SIEMENS

SIMATIC

STEP 7 Basic V11.0 SP1

Online help printout

System overview of STEP 7 and WinCC 1

Readme 2

Installation 3

Migrating projects 4

First steps 5

Introduction to the TIA Portal 6

Editing projects 7

Editing devices and networks 8

Programming a PLC 9

Visualizing processes (Basic) 10

Using technology functions 11

Using online and diagnostics functions 12

Online help printout

08/2011

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠DANGER

indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠WARNING

indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠CAUTION

with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.

CAUTION

without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.

NOTICE

indicates that an unintended result or situation can occur if the relevant information is not taken into account.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠WARNING

Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 08/2011

Table of contents

1 System overview of STEP 7 and WinCC 23

1.1 Scaling of STEP 7 and WinCC 23

1.2 Options for STEP 7 Engineering System 25

1.3 Options for WinCC Engineering and Runtime systems 26

2 Readme 29

2.1 General notes 29
2.1.1 General notes 29
2.1.2 Notes on the installation 33
2.1.3 Use of communications processors 35

2.2 STEP 7 Basic 36
2.2.1 Notes on use 36
2.2.2 Editing devices and networks 37
2.2.2.1 General information on devices and networks 37
2.2.2.2 System reaction of the S7-1200 CPU with firmware version V2.0 37
2.2.2.3 Replacing the ET 200S pulse generator and positioning modules 37
2.2.2.4 Setting flow control for CM 1241 (RS232) 38
2.2.2.5 Notes on online and diagnostics 38
2.2.2.6 Compiling the hardware of a pulse generator 39
2.2.3 Programming a PLC 39
2.2.3.1 General notes on PLC programming 39
2.2.3.2 Instructions 40
2.2.3.3 Testing the user program 41
2.2.4 Technology functions 42
2.2.4.1 Notes on technology functions (S7-1200) 42

2.3 WinCC Readme 43
2.3.1 Notes on use 43
2.3.2 Migration 45
2.3.3 Engineering System 47
2.3.3.1 Screens and Screen Objects 47
2.3.3.2 Tags 50
2.3.3.3 Alarm system and alarm displays 51
2.3.3.4 System functions 51
2.3.3.5 Recipes 51
2.3.3.6 User administration 51
2.3.3.7 Communication 52
2.3.4 Compiling and loading 55
2.3.5 Runtime 58
2.3.5.1 Notes on operation in Runtime 58
2.3.5.2 Notes on operation of panels in Runtime 59
2.3.6 HMI devices 59
2.3.6.1 General notes 59

3 Installation 61

3.1 System requirements for installation ... 61
3.1.1 Notes on the system requirements ... 61
3.1.2 System requirements STEP 7 Basic ... 62
3.1.2.1 Software and hardware requirements STEP 7 ... 62
3.1.3 System requirement for WinCC Basic ... 63
3.1.3.1 Software and hardware requirements ... 63
3.1.3.2 Licenses and Powerpacks ... 66

3.2 Licenses ... 71

3.3 Installation log ... 72

3.4 Starting installation ... 73

3.5 Installing Support Packages ... 76

3.6 Displaying the installed software ... 77

3.7 Modifying or updating installed products ... 78

3.8 Repairing installed products ... 80

3.9 Starting to uninstall ... 82

3.10 Installing and uninstalling the migration tool ... 84
3.10.1 System requirements ... 84
3.10.2 Installing the migration tool ... 84
3.10.3 Uninstalling the migration tool ... 85

4 Migrating projects ... 87

4.1 Migrating projects ... 87

4.2 Preparing projects with the migration tool ... 89
4.2.1 Migrating projects with the migration tool ... 89
4.2.2 Calling the migration tool ... 90
4.2.3 Creating a migration file ... 91

4.3 Migrating projects ... 92

4.4 Displaying the history of the migration ... 94

4.5 Displaying the log file of the migration ... 95

4.6 Migrating STEP 7 projects ... 96
4.6.1 Migration of STEP 7 projects ... 96
4.6.2 Requirements for the migration of STEP 7 projects ... 98
4.6.3 Checking whether STEP 7 projects can be migrated ... 99
4.6.4 Removing unsupported hardware components ... 100
4.6.5 Special points to observe during migration ... 100
4.6.5.1 Device names ... 100
4.6.5.2 PROFINET IO configurations ... 101
4.6.5.3 PROFIBUS configurations ... 102
4.6.5.4 Connections ... 103
4.6.5.5 Multiprojects ... 104
4.6.5.6 Messages ... 104
4.6.5.7 Reporting system errors ... 105
4.6.5.8 Migrating program blocks ... 105

4.7 Migrating WinCC flexible projects (Basic) ... 114
4.7.1 Basics (WinCC flexible) ... 114

4.7.1.1 Migration (WinCC flexible) 114
4.7.1.2 Basics on migration (WinCC flexible) 115
4.7.1.3 Projects from Migrating WinCC flexible projects (WinCC flexible) 117
4.7.1.4 Compiling and loading a migrated project (WinCC flexible) 119
4.7.2 Migrating engineering data (WinCC flexible) 120
4.7.2.1 Supported HMI devices (WinCC flexible) 120
4.7.2.2 Object support during migration (WinCC flexible) 122
4.7.2.3 Changes of values of object properties by the migration (WinCC flexible) 124
4.7.2.4 Migration tags (WinCC flexible) 125
4.7.2.5 Migration of alarm classes and alarm groups (WinCC flexible) 126
4.7.2.6 Migration of language-dependent contents (WinCC flexible) 129
4.7.2.7 Migrating libraries (WinCC flexible) 131
4.7.3 Migrating runtime data (WinCC flexible) 132
4.7.3.1 Migrating runtime data (WinCC flexible) 132
4.7.3.2 Backing up recipe data and user administration (WinCC flexible) 133
4.7.3.3 Restoring recipe data and user administration (WinCC flexible) 134
4.7.4 Migrating integrated projects (WinCC flexible) 135
4.7.4.1 Migrating integrated projects (WinCC flexible) 135
4.7.4.2 Migrating an integrated project (WinCC flexible) 136
4.7.5 Reference (WinCC flexible) 139
4.7.5.1 Migration of data types (WinCC flexible) 139
4.7.5.2 Migrating data types of Allen-Bradley DF1 (WinCC flexible) 140
4.7.5.3 Migrating data types of Allen-Bradley Ethernet IP (WinCC flexible) 141
4.7.5.4 Migrating data types of Mitsubishi FX (WinCC flexible) 142
4.7.5.5 Migrating data types of Modicon Modbus (WinCC flexible) 143
4.7.5.6 Migrating data types of Modicon Modbus TCP/IP (WinCC flexible) 143
4.7.5.7 Migrating data types of Omron Hostlink/Multilink (WinCC flexible) 144
4.7.5.8 Migrating data types of SIMATIC S7 200 (WinCC flexible) 145
4.7.5.9 Migrating data types of SIMATIC S7 300/400 (WinCC flexible) 146

4.8 Post-editing integrated projects 150
4.8.1 Migrating an integrated project 150
4.8.2 Post-editing integrated projects 152
4.8.3 Converting unspecified CPUs into specified CPUs 153
4.8.4 Creating an integrated HMI connection 155
4.8.5 Re-linking HMI tags 157
4.8.6 Deleting an unspecified connection 158

5 First steps 159

5.1 Getting Started Documentation 159

6 Introduction to the TIA Portal 161

6.1 User interface and operation 161
6.1.1 Starting, setting and exiting the TIA Portal 161
6.1.1.1 Starting and exiting the TIA Portal 161
6.1.1.2 Overview of the program settings 162
6.1.1.3 Overview of the script and text editor settings 164
6.1.1.4 Overview of the print settings 165
6.1.1.5 Changing the settings 166
6.1.2 Layout of the user interface 166
6.1.2.1 Views 166
6.1.2.2 Portal view 167
6.1.2.3 Project view 169

6.1.2.4 Project navigation ... 171
6.1.2.5 Work area ... 175
6.1.2.6 Inspector window ... 184
6.1.2.7 Task cards ... 186
6.1.2.8 Details view ... 189
6.1.2.9 Overview window ... 190
6.1.2.10 Resetting the user interface layout ... 193
6.1.3 Keyboard shortcuts ... 194
6.1.3.1 Keyboard shortcuts for project editing ... 194
6.1.3.2 Keyboard shortcuts for windows ... 195
6.1.3.3 Keyboard shortcuts in the project tree ... 195
6.1.3.4 Keyboard shortcuts in tables ... 195
6.1.3.5 Keyboard shortcuts for text editing ... 196
6.1.3.6 Using the on-screen keyboard ... 197
6.1.4 Special features specific to the operating system ... 197
6.1.4.1 Influence of user rights ... 197
6.1.4.2 Expanding user rights ... 198

6.2 Help on the information system ... 200
6.2.1 General remarks on the information system ... 200
6.2.2 Opening the Help system ... 203
6.2.3 Searching the Help system for keywords ... 203
6.2.4 Full-text searches ... 204
6.2.5 Using favorites ... 205
6.2.6 Printing help topics ... 206
6.2.7 Disabling the automatic display of tooltip cascades ... 206
6.2.8 Safety Guidelines ... 207

7 Editing projects ... 209

7.1 The basics of projects ... 209

7.2 Creating and managing projects ... 210
7.2.1 Creating a new project ... 210
7.2.2 Opening projects ... 210
7.2.3 Displaying properties of the project ... 212
7.2.4 Saving projects ... 213
7.2.5 Closing projects ... 213
7.2.6 Deleting projects ... 214
7.2.7 Working with multi-language projects ... 215
7.2.7.1 Project text basics ... 215
7.2.7.2 Select project languages ... 217
7.2.7.3 Setting the editing language ... 217
7.2.7.4 Translating texts directly ... 218
7.2.7.5 Translating texts using reference texts ... 219
7.2.7.6 Exporting and importing project texts ... 220
7.2.7.7 Application examples for multilanguage projects ... 222

7.3 Editing project data ... 224
7.3.1 Compiling and loading project data ... 224
7.3.1.1 Compiling project data ... 224
7.3.1.2 Downloading project data ... 226
7.3.2 Comparing project data ... 233
7.3.2.1 Basics of project data comparison ... 233
7.3.3 Protecting project data ... 234

7.3.3.1 Protection concept for project data 234
7.3.3.2 Revoking access rights for devices....235
7.3.4 Printing project contents 235
7.3.4.1 Documentation settings 235
7.3.4.2 Printout of project contents 237
7.3.4.3 Changing the print settings 238
7.3.4.4 Specifying the print layout....239
7.3.4.5 Entering document information....240
7.3.4.6 Managing cover pages and frames 241
7.3.4.7 Designing cover pages and frames 244
7.3.4.8 Displaying print preview....251
7.3.4.9 Printing project data....254

7.4 Undoing and redoing actions 257
7.4.1 Basics of undoing and redoing actions....257
7.4.2 Undoing an action....259
7.4.3 Redoing an action....260

7.5 Finding and replacing in projects....261
7.5.1 Information on the search function 261
7.5.2 Search and replace....262

7.6 Working with text lists 264
7.6.1 Text lists....264
7.6.2 Creating user-defined text lists 266
7.6.3 Editing user-defined text lists....267
7.6.4 Editing system-defined text lists....267

7.7 Using memory cards....269
7.7.1 Basics about memory cards....269
7.7.2 Adding a user-defined card reader 269
7.7.3 Accessing memory cards....270
7.7.4 Displaying properties of memory cards 271

7.8 Using libraries 272
7.8.1 Library basics....272
7.8.2 "Libraries" task card 273
7.8.3 Using the elements and parts view....276
7.8.4 Working with the project library....276
7.8.4.1 Project library basics....276
7.8.4.2 Creating folders in the project library 277
7.8.4.3 Adding elements to the project library 277
7.8.4.4 Using elements of the project library 279
7.8.4.5 Editing elements of a project library....282
7.8.4.6 Removing elements from the project library 284
7.8.4.7 Filtering the view 285
7.8.5 Working with global libraries 285
7.8.5.1 Global library basics....285
7.8.5.2 Creating a new global library 286
7.8.5.3 Opening a global library....287
7.8.5.4 Displaying properties of global libraries....288
7.8.5.5 Saving a global library....289
7.8.5.6 Closing a global library....290
7.8.5.7 Deleting a global library 291
7.8.5.8 Creating folders in the global libraries 292

7.8.5.9 Adding elements to a global library 293
7.8.5.10 Using elements of a global library 295
7.8.5.11 Editing elements of a global library 298
7.8.5.12 Removing elements from a global library 300
7.8.5.13 Using a supplied global library 301
7.8.5.14 Filtering the view 302

7.9 Using cross-references 303
7.9.1 Using cross-references 303

7.10 Simulating devices 304
7.10.1 Simulation of devices 304
7.10.2 Starting the simulation 304

8 Editing devices and networks 305

8.1 Configuring devices and networks 305
8.1.1 Hardware and network editor 305
8.1.1.1 Overview of hardware and network editor 305
8.1.1.2 Network view 307
8.1.1.3 Device view 310
8.1.1.4 Topology view 313
8.1.1.5 Printing hardware and network configurations 315
8.1.1.6 Activating the page break preview for printout 317
8.1.1.7 Changing the print options 317
8.1.1.8 Inspector window 318
8.1.1.9 Hardware catalog 320
8.1.1.10 Information on hardware components 322
8.1.1.11 Enabling product support 323
8.1.1.12 Keyboard action in the hardware and network editor 324
8.1.2 Configuring devices 325
8.1.2.1 Basics 325
8.1.2.2 Configuring individual devices 334
8.1.3 Configure networks 348
8.1.3.1 Networking devices 348
8.1.3.2 Communication via connections 369
8.1.3.3 Displaying and configuring topology 414
8.1.4 Creating configurations 430
8.1.4.1 Configurations for automation systems 430
8.1.4.2 Configurations for PROFIBUS DP 465
8.1.4.3 Configurations for PROFINET IO 488
8.1.4.4 Bus coupling with PN/PN coupler 519
8.1.4.5 Configurations using external tools 521
8.1.4.6 Loading a configuration 523
8.1.5 Additional information on configurations 525
8.1.5.1 Functional description of S7-1200 CPUs 525
8.1.5.2 Distributed I/O 588
8.1.5.3 Parameters of analog modules 619
8.1.6 System diagnostics with 'Report System Errors' 620
8.1.6.1 Introduction to system diagnostics with 'Report System Errors' 620
8.1.6.2 Basics of system diagnostics 620
8.1.6.3 Components supported 621
8.1.6.4 Diagnostics blocks for reporting system errors 622
8.1.6.5 Properties of the blocks 622

8.1.6.6 Supported error OBs....623
8.1.6.7 Overview of the status DBs....624
8.1.6.8 Diagnostics status DB....625
8.1.6.9 PROFINET IO DB....628
8.1.6.10 PROFIBUS DP DB....631
8.1.6.11 Displaying settings of system diagnostics....633
8.1.6.12 Basic settings....633
8.1.6.13 Advanced settings....637
8.1.7 Displaying alarms....641
8.1.7.1 Overview of the alarm display....641
8.1.7.2 Archive view....642
8.1.7.3 Layout of the alarms in the archive view....642
8.1.7.4 Receiving alarms....643
8.1.7.5 Export archive....643
8.1.7.6 Clear archive....643
8.1.7.7 "Active alarms" view....644
8.1.7.8 Layout of the alarms in the "Active alarms" view....644
8.1.7.9 Status of the alarms....644
8.1.7.10 Acknowledging alarms....645
8.1.7.11 Ignoring alarms....646
8.1.7.12 Keyboard commands in the alarm display....646

8.2 Device and network diagnostics....647
8.2.1 Hardware diagnostics....647
8.2.1.1 Overview of hardware diagnostics....647
8.2.1.2 Showing non-editable and current values of configurable module properties....656
8.2.1.3 Showing the current values of dynamic modules properties....661
8.2.1.4 Checking a module for defects....664
8.2.1.5 Changing the properties of a module or the programming device / PC....670
8.2.1.6 Diagnostics in STOP mode....678
8.2.1.7 Online accesses in the Online and Diagnostics view....680
8.2.1.8 Checking PROFIBUS DP subnets for faults....683
8.2.2 Connection diagnostics....686
8.2.2.1 Overview of connection diagnostics....686
8.2.2.2 Displaying the connection status using icons....687
8.2.2.3 Detailed connection diagnostics....688

9 Programming a PLC....693

9.1 Creating a user program....693
9.1.1 Programming basics....693
9.1.1.1 Operating system and user program....693
9.1.1.2 Blocks in the user program....694
9.1.1.3 Block calls....702
9.1.1.4 Operands in instructions....716
9.1.1.5 Data types....741
9.1.1.6 Program flow control....819
9.1.2 Declaring PLC tags....824
9.1.2.1 Overview of PLC tag tables....824
9.1.2.2 Structure of the PLC tag tables....825
9.1.2.3 Creating and managing PLC tag tables....826
9.1.2.4 Declaring PLC tags....829
9.1.2.5 Declaring symbolic constants....835
9.1.2.6 Editing properties....837

9.1.2.7 Monitoring of PLC tags 840
9.1.2.8 Editing PLC tag tables 840
9.1.3 Creating and managing blocks 844
9.1.3.1 Creating blocks 844
9.1.3.2 Specifying block properties 853
9.1.3.3 Managing blocks 859
9.1.4 Programming blocks 863
9.1.4.1 Program editor 863
9.1.4.2 Programming code blocks 875
9.1.4.3 Programming data blocks 1014
9.1.4.4 Programming PLC data types 1036
9.1.4.5 Using external source files 1047
9.1.5 Comparing project data 1052
9.1.5.1 Basics of comparing project data 1052
9.1.5.2 Carrying out an online/offline comparison 1056
9.1.5.3 Carrying out offline/offline comparisons 1057
9.1.5.4 Using the comparison editor 1059
9.1.5.5 Performing detailed block comparisons 1065
9.1.6 Compiling and downloading blocks 1071
9.1.6.1 Compiling blocks 1071
9.1.6.2 Downloading blocks 1076
9.1.7 Protecting blocks 1085
9.1.7.1 Protecting blocks 1085
9.1.7.2 Setting up and removing block copy protection 1087
9.1.7.3 Setting up block know-how protection 1088
9.1.7.4 Opening know-how protected blocks 1089
9.1.7.5 Printing know-how protected blocks 1090
9.1.7.6 Changing a password 1091
9.1.7.7 Removing block know-how protection 1092

9.2 Displaying program information 1093
9.2.1 Overview of available program information 1093
9.2.2 Displaying an assignment list 1094
9.2.2.1 Introduction to the assignment list 1094
9.2.2.2 Layout of the assignment list 1095
9.2.2.3 Symbols in the assignment list 1096
9.2.2.4 Displaying an assignment list 1097
9.2.2.5 Setting the view options for the assignment list 1098
9.2.2.6 Filter options in the assignment list 1099
9.2.2.7 Defining filters for assignment list 1100
9.2.2.8 Filtering an assignment list 1101
9.2.2.9 Defining retentive memory areas for bit memories 1101
9.2.2.10 Enabling the display of retentive bit memories 1102
9.2.3 Displaying the call structure 1103
9.2.3.1 Introduction to the call structure 1103
9.2.3.2 Symbols in the call structure 1105
9.2.3.3 Layout of the call structure 1106
9.2.3.4 Displaying the call structure 1107
9.2.3.5 Setting the view options for the call structure 1108
9.2.3.6 Introducing the consistency check in the call structure 1109
9.2.3.7 Checking block consistency in the call structure 1110
9.2.4 Displaying the dependency structure 1111
9.2.4.1 Introduction to the dependency structure 1111

9.2.4.2 Layout of the dependency structure 1112
9.2.4.3 Symbols in the dependency structure....1113
9.2.4.4 Displaying the dependency structure....1113
9.2.4.5 Setting the view options for the dependency structure 1114
9.2.4.6 Introducing the consistency check in the dependency structure 1115
9.2.4.7 Checking block consistency in the dependency structure 1116
9.2.5 Displaying CPU resources 1117
9.2.5.1 Introducing resources 1117
9.2.5.2 Layout of the "Resources" tab 1119
9.2.5.3 Displaying resources....1120
9.2.5.4 Selecting the maximum load memory available 1120

9.3 Displaying cross-references....1121
9.3.1 General information about cross references....1121
9.3.2 Structure of the cross-reference list 1122
9.3.3 Displaying the cross-reference list 1124
9.3.4 Displaying cross-references in the Inspector window....1125

9.4 Testing the user program....1127
9.4.1 Basics of testing the user program 1127
9.4.2 Testing with program status....1128
9.4.2.1 Testing the program....1128
9.4.3 Testing with the watch table....1133
9.4.3.1 Introduction to testing with the watch table....1133
9.4.3.2 Layout of the watch table 1134
9.4.3.3 Basic mode and expanded mode in the watch table 1135
9.4.3.4 Icons in the watch table 1136
9.4.3.5 Creating and editing watch tables....1137
9.4.3.6 Entering tags in the watch table....1140
9.4.3.7 Monitoring tags in the watch table 1145
9.4.3.8 Modifying tags in the watch table....1149
9.4.4 Testing with the force table 1156
9.4.4.1 Introduction for testing with the force table 1156
9.4.4.2 Safety precautions when forcing tags....1158
9.4.4.3 Layout of the force table 1159
9.4.4.4 Basic mode and expanded mode in the force table 1160
9.4.4.5 Icons in the force table....1160
9.4.4.6 Open and edit force table....1161
9.4.4.7 Entering tags in the force table 1162
9.4.4.8 Monitoring tags in the force table....1167
9.4.4.9 Forcing tags in the force table....1170
9.4.4.10 Stop forcing tags 1178

9.5 Configuring alarms 1181
9.5.1 Introduction to alarm configuration 1181
9.5.2 Assigning alarm numbers 1181
9.5.3 Components of an alarm....1182
9.5.4 Available alarm blocks 1182
9.5.5 Alarm type and alarms 1183
9.5.6 Formal parameters, alarm data types and alarm blocks 1185
9.5.7 Layout of the alarm editor 1186
9.5.8 Creating and editing alarms 1187
9.5.8.1 Creating PLC alarms....1187
9.5.8.2 Edit PLC alarms in the alarm editor 1188

9.5.8.3 Edit PLC alarms in the program editor ... 1188
9.5.8.4 Deleting PLC alarms ... 1189
9.5.8.5 Creating and editing an alarm type ... 1189
9.5.8.6 Creating and editing an instance DB ... 1189
9.5.8.7 Creating user diagnostic messages ... 1190
9.5.8.8 Editing user diagnostic alarms ... 1191
9.5.8.9 Deleting user diagnostic messages ... 1191
9.5.8.10 Entering texts ... 1191
9.5.8.11 Locking texts ... 1192
9.5.8.12 Locking attributes ... 1193
9.5.8.13 Insert associated values in alarms ... 1193
9.5.8.14 Structure of associated values ... 1194
9.5.8.15 Examples of associated values ... 1195
9.5.8.16 Deleting associated values ... 1196
9.5.9 Text lists for alarms ... 1196
9.5.9.1 Basics of alarm text lists ... 1196
9.5.9.2 Editing text lists for alarms ... 1196
9.5.9.3 Integrating texts from text lists in alarms ... 1197
9.5.9.4 Example of integrating texts from text lists in alarms ... 1198
9.5.10 Alarm classes ... 1198
9.5.10.1 Creating alarm classes ... 1198
9.5.10.2 Editing alarm classes ... 1199

9.6 Using global project functions ... 1200
9.6.1 Importing and exporting ... 1200
9.6.1.1 Basics for importing and exporting ... 1200
9.6.1.2 Format of the export file ... 1200
9.6.1.3 Exporting PLC tags ... 1201
9.6.1.4 Importing PLC tags ... 1202

9.7 Programming examples ... 1203
9.7.1 LAD programming examples ... 1203
9.7.1.1 Example of controlling a conveyor belt ... 1203
9.7.1.2 Example of detecting the direction of a conveyor belt ... 1204
9.7.1.3 Example of detecting the fill level of a storage area ... 1206
9.7.1.4 Example of controlling room temperature ... 1209
9.7.2 FBD programming examples ... 1211
9.7.2.1 Example of controlling a conveyor belt ... 1211
9.7.2.2 Example of detecting the direction of a conveyor belt ... 1212
9.7.2.3 Example of detecting the fill level of a storage area ... 1214
9.7.2.4 Example of controlling room temperature ... 1217

9.8 References ... 1220
9.8.1 General parameters of the instructions ... 1220
9.8.1.1 Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions ... 1220
9.8.1.2 Evaluating errors with output parameter RET_VAL ... 1222
9.8.2 Basic instructions ... 1226
9.8.2.1 LAD ... 1226
9.8.2.2 FBD ... 1393
9.8.2.3 SCL ... 1565
9.8.3 Extended instructions ... 1668
9.8.3.1 Date and time-of-day ... 1668
9.8.3.2 String + Char ... 1678
9.8.3.3 Distributed I/O ... 1700

9.8.3.4 Interrupts 1729
9.8.3.5 Diagnostics 1745
9.8.3.6 Pulse 1755
9.8.3.7 Data logging 1757
9.8.3.8 Data block functions 1767
9.8.4 Technology 1771
9.8.4.1 PID Control 1771
9.8.4.2 Motion Control 1816
9.8.4.3 High-speed counters 1848
9.8.5 Communication 1850
9.8.5.1 S7 communication 1850
9.8.5.2 Open User Communication 1859
9.8.5.3 Web server 1893
9.8.5.4 Communications processor 1895
9.8.5.5 TeleService 1958

10 Visualizing processes (Basic) 1967

10.1 Creating screens 1967
10.1.1 Basics 1967
10.1.1.1 Screen basics 1967
10.1.1.2 Availability of screens for specific HMI devices 1969
10.1.1.3 Basics 1970
10.1.1.4 Working with screens 1974
10.1.1.5 Working with Templates 1977
10.1.2 Working with objects 1983
10.1.2.1 Overview of objects 1983
10.1.2.2 Options for Editing Objects 1986
10.1.2.3 Inserting an object 1987
10.1.2.4 Deleting an Object 1989
10.1.2.5 Positioning an object 1990
10.1.2.6 Resizing an object 1991
10.1.2.7 Selecting multiple objects 1992
10.1.2.8 Aligning objects 1995
10.1.2.9 Moving an object forward or backward 1997
10.1.2.10 Show objects outside the screen area 1998
10.1.2.11 Rotating objects 1998
10.1.2.12 Flipping objects 2000
10.1.2.13 Designing an object 2001
10.1.2.14 Inserting multiple objects of the same type (stamping tool) 2001
10.1.2.15 Repositioning and resizing multiple objects 2003
10.1.2.16 External graphics 2004
10.1.2.17 Managing external graphics 2005
10.1.2.18 Storing an external image in the graphics library. 2007
10.1.2.19 Working with object groups 2009
10.1.2.20 Configuring the keyboard access 2015
10.1.2.21 Examples 2019
10.1.3 Working with text lists and graphics lists 2022
10.1.3.1 Working with text lists 2022
10.1.3.2 Working with graphics lists 2029
10.1.4 Dynamizing screens 2038
10.1.4.1 Basics on dynamization 2038
10.1.4.2 Dynamization in the inspector window 2038

10.1.4.3 Dynamization with animations 2040
10.1.4.4 Dynamize with system functions 2049
10.1.5 Working with function keys 2051
10.1.5.1 Working with function keys 2051
10.1.5.2 Assigning function keys globally 2053
10.1.5.3 Local assignment of function keys 2055
10.1.5.4 Assigning a function key to a function 2057
10.1.5.5 Assigning operator authorization for a function key 2058
10.1.5.6 Assigning a function key to a graphic 2060
10.1.5.7 Example: Using function keys for screen navigation 2062
10.1.6 Working with layers 2064
10.1.6.1 Basics on working with layers 2064
10.1.6.2 Moving objects between layers 2065
10.1.6.3 Setting the active layer 2066
10.1.6.4 Show and hide layers 2067
10.1.6.5 Renaming layers 2068
10.1.7 Working with libraries 2070
10.1.7.1 Basics on libraries 2070
10.1.7.2 Copy templates and types 2072
10.1.7.3 Libraries in WinCC 2072
10.1.7.4 Displaying library objects 2075
10.1.7.5 Managing library objects 2077
10.1.7.6 Storing an object in a library 2078
10.1.7.7 Inserting a library object 2079
10.1.7.8 Creating a global library 2080
10.1.7.9 Saving a global library 2081
10.1.7.10 Opening a global library 2082
10.1.8 Display and operating objects 2083
10.1.8.1 Device-Specific Nature of the Objects 2083
10.1.8.2 Objects 2084
10.1.9 Configuring screen navigation 2111
10.1.9.1 Basics for screen navigation 2111
10.1.9.2 Assign button with screen change 2112
10.1.9.3 Assign screen change to function key 2114

10.2 Working with Tags 2115
10.2.1 Basics 2115
10.2.1.1 Basics of tags 2115
10.2.1.2 Overview of HMI tag tables 2116
10.2.1.3 External tags 2117
10.2.1.4 Addressing external tags 2118
10.2.1.5 Internal tags 2121
10.2.2 Working with tags 2122
10.2.2.1 Creating tags 2122
10.2.2.2 Editing tags 2127
10.2.2.3 Configuring tags 2132
10.2.3 Working with arrays 2150
10.2.3.1 Basics on arrays 2150
10.2.3.2 Creating array tags 2152
10.2.3.3 Examples of arrays 2153
10.2.4 Working with cycles 2154
10.2.4.1 Cycle basics 2154
10.2.5 Displaying tags 2155

10.2.5.1 Outputting tag values in screens....2155
10.2.5.2 Configuring trend displays for values from the PLC 2155

10.3 Working with alarms....2158
10.3.1 Basics....2158
10.3.1.1 Alarm system in WinCC 2158
10.3.1.2 Alarm types 2159
10.3.1.3 Alarm states 2161
10.3.1.4 Alarm classes....2162
10.3.1.5 Acknowledgment....2164
10.3.1.6 Alarm groups....2165
10.3.1.7 Alarm number....2166
10.3.2 Working with alarms....2167
10.3.2.1 Alarm components and properties....2167
10.3.2.2 Configuring alarms 2168
10.3.2.3 Configuring alarm output....2182
10.3.2.4 Acknowledging alarms 2187
10.3.3 Operating alarms in Runtime 2190
10.3.3.1 Alarms in Runtime....2190
10.3.3.2 Simple alarm view, simple alarm window in runtime 2192
10.3.3.3 Alarm indicator in Runtime....2194
10.3.3.4 Acknowledging alarms 2194
10.3.4 Reference....2195
10.3.4.1 System functions for alarms....2195
10.3.4.2 System events....2196

10.4 Working with recipes 2219
10.4.1 Basics....2219
10.4.1.1 Definition and applications 2219
10.4.1.2 Examples for using recipes....2221
10.4.1.3 Structure of recipes 2221
10.4.1.4 Displaying recipes 2223
10.4.1.5 Flow of data for recipes....2223
10.4.1.6 Synchronization of recipe data records with the PLC 2224
10.4.2 Elements and basic settings 2225
10.4.2.1 "Recipes" editor....2225
10.4.3 Displaying and editing recipes in Runtime....2228
10.4.3.1 Simple recipe view 2228
10.4.3.2 Behavior of the recipe view in Runtime 2229
10.4.4 Configuring recipes 2231
10.4.4.1 General configuration procedure 2231
10.4.4.2 Creating and Editing Recipes 2232
10.4.4.3 Configuring the display of recipes....2239
10.4.5 Using recipes in Runtime 2242
10.4.5.1 Using the simple recipe view 2242
10.4.6 Example 2248
10.4.6.1 Example of creating a recipe 2248

10.5 Configuring user administration 2251
10.5.1 Field of application of the user administration 2251
10.5.2 Form of the user administration 2252
10.5.3 Basics....2253
10.5.3.1 Users....2253
10.5.3.2 Users work area....2253

10.5.3.3 User groups 2254
10.5.3.4 User groups work area 2255
10.5.3.5 Settings for the user administration 2256
10.5.4 Setting up the user administration 2258
10.5.4.1 Basics on user administration 2258
10.5.4.2 Administering users for Runtime 2258
10.5.4.3 Managing users in Runtime 2265
10.5.4.4 Configuring access protection 2273
10.5.5 Reference 2274
10.5.5.1 Objects with access protection 2274
10.5.5.2 Default user groups and authorizations 2274
10.5.6 Examples 2275
10.5.6.1 Example: Configuring a button with logon dialog box 2275
10.5.6.2 Example: Logging the logon and logoff events 2276
10.5.6.3 Example of user management 2277

10.6 Working with system functions 2285
10.6.1 Basics 2285
10.6.1.1 System functions 2285
10.6.1.2 Use of system functions 2286
10.6.2 Working with function lists 2287
10.6.2.1 Basic of the functions list 2287
10.6.2.2 Properties of a function list 2288
10.6.2.3 Configuring a function list 2288
10.6.2.4 Editing a function list 2290
10.6.2.5 Executing a function list in Runtime 2291
10.6.3 Example 2291
10.6.3.1 Changing the operating mode on the HMI device with the current display 2291
10.6.4 Reference 2294
10.6.4.1 Function list 2294
10.6.4.2 Events 2354

10.7 Planning tasks 2373
10.7.1 Field of application of the Scheduler 2373
10.7.2 Working with tasks and triggers 2374
10.7.3 Basics 2375
10.7.3.1 Work area of the "Scheduler" editor 2375
10.7.3.2 Function list 2376
10.7.3.3 Triggers 2377
10.7.3.4 Planning tasks with event triggers 2377
10.7.4 Examples 2379
10.7.4.1 Example: Update user following change of user 2379

10.8 Communicating with PLCs 2381
10.8.1 Basics of communication 2381
10.8.1.1 Communication between devices 2381
10.8.1.2 Devices and networks in the automation system 2382
10.8.1.3 Data exchange using tags 2385
10.8.1.4 Data exchange using area pointers 2386
10.8.1.5 Communication drivers 2386
10.8.2 Networks and connections 2387
10.8.2.1 SIMATIC communication networks 2387
10.8.2.2 Configuring networks and connections 2391
10.8.3 Data exchange 2399

10.8.3.1 Data exchange using tags 2399
10.8.3.2 Data exchange using area pointers 2406
10.8.4 Device dependency 2413
10.8.4.1 Basic Panel 2413
10.8.5 Communicating with SIMATIC S7 1200 2417
10.8.5.1 Communication with SIMATIC S7 1200 2417
10.8.5.2 Communication via PROFINET 2418
10.8.5.3 Communication via PROFIBUS 2432
10.8.5.4 Data exchange 2441
10.8.5.5 Performance features of communication 2460
10.8.5.6 Creating connections in the "Connections" editor 2461
10.8.6 Communicating with SIMATIC S7 300/400 2470
10.8.6.1 Communication with SIMATIC S7 300/400 2470
10.8.6.2 Communication via PROFINET 2471
10.8.6.3 Communication via PROFIBUS 2484
10.8.6.4 Communication via MPI 2492
10.8.6.5 Data exchange 2502
10.8.6.6 Performance features of communication 2519
10.8.6.7 Creating connections in the "Connections" editor 2520
10.8.7 Communicating with SIMATIC S7 200 2532
10.8.7.1 Communication with SIMATIC S7 200 2532
10.8.7.2 Creating a connection to SIMATIC S7 200 2532
10.8.7.3 Parameters for the connection 2534
10.8.7.4 Data exchange 2542
10.8.7.5 Performance features of communication 2560
10.8.8 Communication with other PLCs 2561
10.8.8.1 Communication with other PLCs 2561
10.8.8.2 Distinctive features when configuring 2562
10.8.8.3 Communication drivers 2562
10.8.8.4 Data exchange using area pointers 2676

10.9 Using global functions 2692
10.9.1 HMI device wizard basics 2692
10.9.2 Importing and exporting project data 2693
10.9.2.1 Importing and exporting project data 2693
10.9.2.2 Importing and exporting recipes 2694
10.9.2.3 Importing and exporting alarms 2698
10.9.2.4 Importing and exporting tags 2706
10.9.2.5 Importing and exporting text lists 2711
10.9.2.6 Importing and exporting project texts 2716
10.9.3 Using cross-references 2719
10.9.3.1 General information about cross references 2719
10.9.3.2 Displaying the cross-reference list 2719
10.9.3.3 Structure of the cross-reference list 2720
10.9.3.4 Displaying cross-references in the Inspector window 2721
10.9.4 Managing languages 2723
10.9.4.1 Languages in WinCC 2723
10.9.4.2 Language settings in the operating system 2724
10.9.4.3 Operating system settings for Asian languages 2725
10.9.4.4 Setting project languages 2726
10.9.4.5 Creating one project in multiple languages 2729
10.9.4.6 Using language-specific graphics 2737
10.9.4.7 Languages in runtime 2741

10.9.4.8 Example of multilingual configuration 2748
10.9.5 Replacing devices 2751
10.9.5.1 Basics 2751
10.9.5.2 Device-specific functions 2752
10.9.5.3 Adjusting screens to the new device 2755
10.9.6 Copying between devices and editors 2759
10.9.6.1 Basics 2759
10.9.6.2 Copying and pasting 2761

10.10 Compiling and loading 2766
10.10.1 Compiling and loading projects 2766
10.10.1.1 Overview of compiling and loading projects 2766
10.10.1.2 Compiling a project 2767
10.10.1.3 Loading projects 2768
10.10.1.4 Runtime start 2771
10.10.2 Simulating projects 2772
10.10.2.1 Simulation basics 2772
10.10.2.2 Simulating a project 2773
10.10.2.3 Working with the tag simulator 2775
10.10.3 Servicing the HMI device 2777
10.10.3.1 ProSave 2777
10.10.3.2 Backup of HMI data 2778
10.10.3.3 Backing up and restoring data of the HMI device 2779
10.10.3.4 Updating the operating system 2780
10.10.3.5 Updating the HMI device operating system 2781
10.10.4 Reference 2782
10.10.4.1 Error messages during loading of projects 2782
10.10.4.2 Adapting the project for another HMI device 2784
10.10.4.3 Establishing a connection to the HMI device 2785

10.11 Operating in Runtime 2787
10.11.1 Basics 2787
10.11.1.1 Overview 2787
10.11.1.2 Tags in Runtime 2788
10.11.1.3 System functions in Runtime 2788
10.11.2 Commissioning projects 2789
10.11.2.1 Settings in the Runtime software 2789
10.11.2.2 Loading projects 2790
10.11.2.3 Starting Runtime on the Engineering Station 2791
10.11.2.4 Starting Runtime on the HMI device 2792
10.11.2.5 Testing a project 2792
10.11.2.6 Closing a project 2793
10.11.2.7 Backing up and restoring data of the HMI device 2794
10.11.3 Languages in runtime 2795
10.11.3.1 Languages in runtime 2795
10.11.3.2 Setting a runtime language 2795
10.11.3.3 Setting the font for a runtime language 2797
10.11.3.4 Configuring language switching 2798
10.11.3.5 Specific features of Asian and Eastern languages in runtime 2798
10.11.4 Operating projects 2799
10.11.4.1 Basics 2799
10.11.4.2 Operating objects 2808
10.11.4.3 Project security 2821

10.11.4.4 Operating alarms 2829
10.11.4.5 Operating recipes....................2837

10.12 Performance features 2848
10.12.1 Engineering system 2848
10.12.2 Basic Panel 2850
10.12.3 General technical data 2854
10.12.3.1 Recommended printers 2854
10.12.3.2 Memory requirements of recipes for Basic Panels 2854

10.13 Migration to WinCC V11 2856
10.13.1 Overview of migration to WinCC V11 2856
10.13.2 Libraries 2856
10.13.3 Screens and templates 2858

11 Using technology functions.................... 2861

11.1 PID control 2861
11.1.1 Principles for control 2861
11.1.1.1 Controlled system and actuators 2861
11.1.1.2 Controlled systems 2863
11.1.1.3 Characteristic values of the control section 2865
11.1.1.4 Pulse controller 2868
11.1.1.5 Response to setpoint changes and disturbances 2871
11.1.1.6 Control Response at Different Feedback Structures 2872
11.1.1.7 Selection of the controller structure for specified controlled systems....................2879
11.1.1.8 PID parameter settings 2880
11.1.2 Configuring a software controller 2881
11.1.2.1 Steps for the configuration of a software controller 2881
11.1.2.2 Add technology objects....................2882
11.1.2.3 Configure technology objects....................2883
11.1.2.4 Call instruction in the user program 2884
11.1.2.5 Commissioning software controller....................2885
11.1.2.6 Loading to CPU....................2885
11.1.2.7 Save optimized PID parameter in the project 2886
11.1.2.8 Display instance DB of a technology object....................2887
11.1.3 PID control (S7-1200) 2887
11.1.3.1 Using PID_Compact....................2887
11.1.3.2 Using PID_3Step....................2907

11.2 Motion Control....................2928
11.2.1 Motion Control (S7-1200)....................2928
11.2.1.1 Using S7-1200 Motion Control....................2928

12 Using online and diagnostics functions.................... 3065

12.1 General information about online mode....................3065

12.2 View in online mode....................3067

12.3 Online access....................3069

12.4 Displaying accessible devices 3071

12.5 Opening the properties of an interface 3073

12.6 Establishing and canceling an online connection 3074

12.7 Connecting online with several devices 3076

12.8 Basics of assigning parameters for the PG/PC interface 3077

12.9 Adding interfaces 3078

12.10 Setting parameters for the Ethernet interface 3079
12.10.1 Setting parameters for the Industrial Ethernet interface 3079
12.10.2 Displaying operating system parameters 3080
12.10.3 Connecting the PG/PC interface to a subnet 3081
12.10.4 Setting parameters for the Ethernet interface 3081
12.10.5 Assigning a temporary IP address 3083
12.10.6 Managing temporary IP addresses 3083
12.10.7 Resetting the TCP/IP configuration 3084

12.11 Setting parameters for the MPI and PROFIBUS interfaces 3085
12.11.1 Setting parameters for the MPI and PROFIBUS interfaces 3085
12.11.2 Setting MPI or PROFIBUS interface parameters automatically 3086
12.11.3 Setting parameters for the MPI interface 3087
12.11.4 Setting parameters for the PROFIBUS interface 3089
12.11.5 Overview of the bus parameters for PROFIBUS 3091
12.11.6 Resetting the MPI or PROFIBUS configuration 3093

12.12 Establishing a remote connection with TeleService 3094
12.12.1 Basics of working with TeleService 3094
12.12.1.1 Introduction to TeleService 3094
12.12.1.2 TeleService functionality 3095
12.12.1.3 Telephone book at TeleService 3096
12.12.2 Telephone networks and modems 3096
12.12.2.1 Supported telephone networks and modems 3096
12.12.2.2 Installing the local modem 3097
12.12.2.3 Connecting and configuring the remote modem 3098
12.12.2.4 Initialization string requirements 3099
12.12.3 Using a TS adapter to establish a remote connection 3100
12.12.3.1 Basics of using a TS adapter 3100
12.12.3.2 Installing TS adapter software 3101
12.12.3.3 TS adapter MPI 3101
12.12.3.4 TS adapter IE 3109
12.12.4 Access protection for TeleService and the TS Adapter 3115
12.12.4.1 Access protection information 3115
12.12.4.2 TeleService callback options 3115
12.12.4.3 Levels of protection 3116
12.12.4.4 Setting up access protection and callback number for the TS adapter 3117
12.12.4.5 Complete a callback in TeleService 3118
12.12.5 Establishing a remote connection to a remote plant 3119
12.12.5.1 Establishing a remote connection 3119
12.12.5.2 Terminating a remote connection 3121
12.12.5.3 Checklist for troubleshooting the modem 3121
12.12.6 Working with the phone book 3122
12.12.6.1 Basics on working with the phone book 3122
12.12.6.2 Structure of the phone book 3123
12.12.6.3 Symbols in the phone book 3124
12.12.6.4 Manage phone book 3124
12.12.7 CPU controlled TeleService remote connections 3126
12.12.7.1 Overview of CPU controlled remote connections 3126
12.12.7.2 Establishing a connection from and to remote systens (PG-AS-remote coupling) 3127

12.12.7.3 Data exchange between remote systems (AS-AS-remote coupling) 3129
12.12.7.4 Send SMS from a system 3131
12.12.7.5 Send an email from a system 3132
12.12.8 Troubleshooting 3134
12.12.8.1 General information on troubleshooting..3134
12.12.8.2 Recording a log file for the modem...3135
12.12.8.3 Remote connection to the TS adapter is not established 3136
12.12.8.4 Remote connection from the TS adapter is not established................................3137
12.12.8.5 Modem connection is interrupted...3138
12.12.8.6 Modem alarms 3139

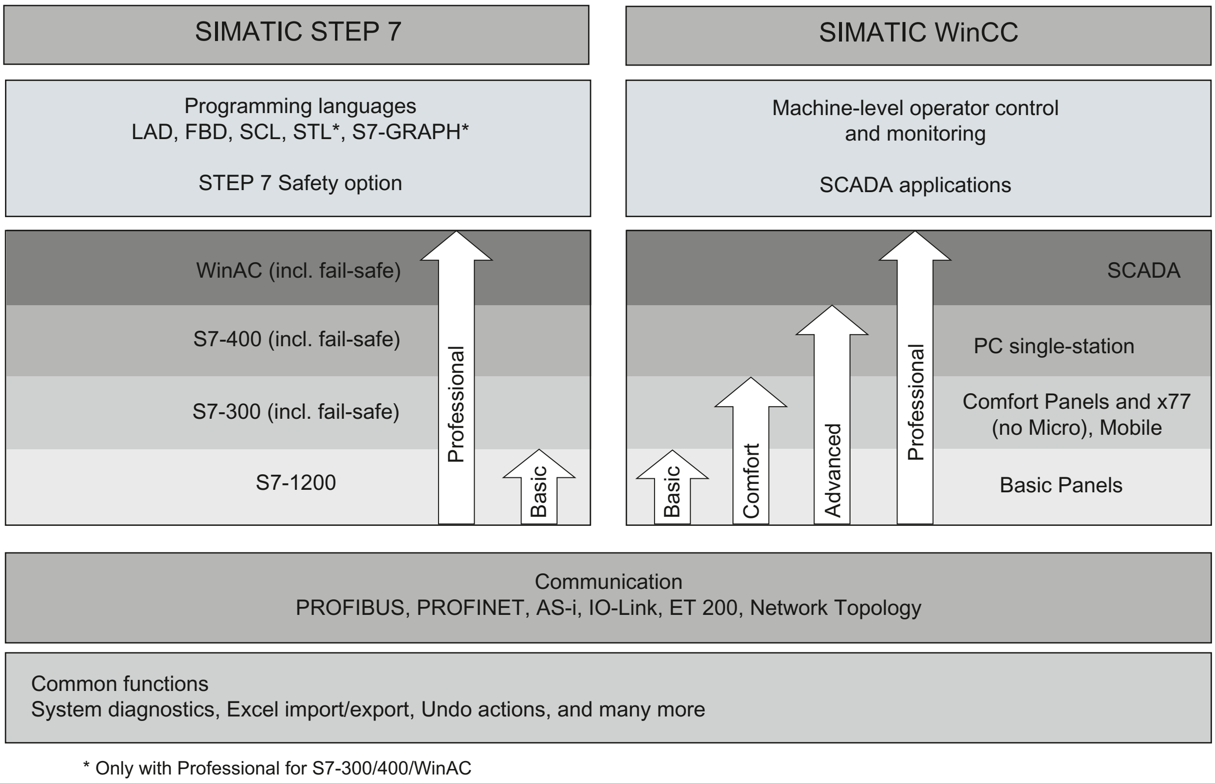
Index... **3141**

System overview of STEP 7 and WinCC 1

1.1 Scaling of STEP 7 and WinCC

Scope of performance of the products

The following graphic shows the scope of performance of the individual products of STEP 7 and WinCC:



STEP 7

STEP 7 (TIA Portal) is the engineering software for configuring the SIMATIC S7-1200, S7-300/400, and WinAC controller families. STEP 7 (TIA Portal) is available in 2 editions, depending on the configurable controller families:

- STEP 7 Basic for configuring the S7-1200
- STEP 7 Professional for configuring S7-1200, S7-300/400, and WinAC

WinCC

WinCC (TIA Portal) is an engineering software for configuring SIMATIC Panels, SIMATIC Industrial PCs, and Standard PCs with the WinCC Runtime Advanced or the SCADA System WinCC Runtime Professional visualization software.

WinCC (TIA Portal) is available in 4 editions, depending on the configurable operator control systems:

- WinCC Basic for configuring Basic Panels

  WinCC Basic is included with every STEP 7 Basic and STEP 7 Professional product.
- WinCC Comfort for configuring all panels (including Comfort Panels, Mobile Panels)
- WinCC Advanced for configuring all panels and PCs with the WinCC Runtime Advanced visualization software

  WinCC Runtime Advanced is a visualization software for PC-based single-station systems. WinCC Runtime Advanced can be purchased with licenses for 128, 512, 2k, 4k as well as 8k PowerTags (tags with a process interface).
- WinCC Professional for configuring panels and PCs with WinCC Runtime Advanced or SCADA System WinCC Runtime Professional

  WinCC Runtime Professional is a SCADA system for structuring a configuration ranging from single-station systems to multi-station systems including standard clients or web clients. WinCC Runtime Professional can be purchased with licenses for 128, 512, 2k, 4k, 8k, 64k, 102400, 153600 as well as 262144 PowerTags (tags with a process interface).

With WinCC (TIA Portal), it is also possible to configure a SINUMERIK PC with WinCC Runtime Advanced or WinCC Runtime Professional and HMI devices with SINUMERIK HMI Pro sl RT or SINUMERIK Operate WinCC RT Basic.

1.2 Options for STEP 7 Engineering System

Additional STEP 7 products

For applications with increased safety requirements, STEP 7 Professional can be extended with the STEP 7 Safety option.

The STEP 7 Safety option allows you to configure fail-safe PLC programs for F-CPUs.

1.3 Options for WinCC Engineering and Runtime systems

SIMATIC Panels as well as WinCC Runtime Advanced and WinCC Runtime Professional contain all essential functions for operator control and monitoring of machines or plants. Additional options allow you to extend the functionality in some cases to increase the range of available tasks.

Options for Comfort Panels, Mobile Panels, Multi Panels

The following possible extensions are available for Comfort Panels, Mobile Panels, and Multi Panels:

- WinCC SmartServer (remote operation)
- WinCC Audit (audit trail and electronic signature for regulated applications)

Note

In contrast to WinCC flexible 2008, functions from the WinCC flexible /Sm@rtService, WinCC flexible /Sm@rtAccess options as well as the WinCC flexible /OPC Server option are incorporated into the basic functionality.

Options for WinCC Runtime Advanced

The following possible extensions are available for WinCC Runtime Advanced:

- WinCC SmartServer (remote operation)
- WinCC Recipes (recipe system)
- WinCC Logging (logging of process values and alarms)
- WinCC Audit (audit trail for regulated applications)

Note

In contrast to WinCC flexible 2008, functions from the WinCC flexible /Sm@rtService, WinCC flexible /Sm@rtAccess options as well as the WinCC flexible /OPC Server option are incorporated into the basic functionality.

Options for WinCC Runtime Professional

The following possible extensions are available for WinCC Runtime Professional:

- WinCC Client (standard client for structuring multi-station systems)
- WinCC Server (supplements WinCC Runtime to include server functionality)
- WinCC Recipes (recipe system, formerly WinCC /UserArchives)
- WinCC WebNavigator (Web-based operator control and monitoring)
- WinCC DataMonitor (display and evaluation of process states and historical data)

Note

In contrast to WinCC V7, functions from the WinCC /OPC-Server and WinCC /ConnectivityPack options are incorporated into the basic functionality. Likewise, the basic functionality includes the Runtime API from WinCC /ODK.

Beyond the Runtime options, WinCC Runtime Advanced and WinCC Runtime Professional can be extended through customer-specific controls. To develop controls, the WinCC ControlDevelopment option is required.

Readme 2

2.1 General notes

2.1.1 General notes

The information in this readme file supersedes statements made in other documents.

Read the following notes carefully because they include important information for installation and use. Read these notes prior to installation.

Security settings

In order to operate the software packages in the TIA Portal, port 4410 for TCP must be entered as an exception in the Windows firewall during installation.

Installing new .Net versions or .Net service packs

- Close the TIA portal before installing a new .Net version or a new .Net service pack on your programming device/PC.
- Restart the TIA portal only after successful installation of the new .Net version or the new .Net service pack.

Notes on handling

- If a project in the list of projects last used is located on a network drive that is not connected, you may experience delays when opening the "Project" menu.
- When you insert a CPU, you may need to wait for some time if the project editor is open at the same time. This generally takes longer when you insert the first CPU in a newly created project. To be able to continue working more quickly, you should close the project editor before inserting a CPU.
- The message "Application is not responding" may appear in Windows 7 with functions that take a long time to run (loading the CPU for example). If this occurs, wait until the function has correctly finished.
- If you have installed a Microsoft mouse with IntelliPoint, you may find that it superimposes components over the buttons of the title bar. If this is the case, uninstall the IntelliPoint software from Microsoft.
- Enabling the "Virtual Desktop" options with NVIDIA graphics cards can cause problems. In this case, disable the "nView virtual desktop manager" of your NVIDIA graphics driver.

Using the TIA Portal via a remote desktop

In principle, it is possible to use the TIA Portal via a remote desktop connection. During configuration, you should, however, avoid disconnecting the connection to the desktop client. In rare cases, this can lead to the software user interface being blocked.

If you experience this blockage, follow these steps on the desktop client.

1. Open the Windows Task-Manager and close the "rdpclip.exe" process.
2. Type in "rdpclip.exe" in the command prompt to restart the process.

Note that the current content of the clipboard will be lost. You can, however, then continue configuration as usual. To be on the safe side, you should restart the TIA Portal at the next opportunity.

Opening the TIA portal multiple times

If you are running several applications of the TIA portal and they continually become active in turn, you can briefly switch to another application or use the key combination <ALT+Tab> to solve the problem.

Note on SD cards

The SD cards have been formatted and initialized by Siemens for use with S7-1200 modules. This format must not be overwritten otherwise the card will no longer be accepted by the S7-1200 modules. Formatting with Windows tools is therefore not permitted.

Behavior in case of open force job

Note that active force jobs will be retained even after you have loaded a new project to the SD card. This means you should first delete the active force job before you remove an SD card from the CPU and before you overwrite the card in the PC with a new project. If you use an SD card with unknown content, you should format the SD card before the next download.

Issues when shutting down Windows XP or when activating a screen saver

Windows XP uses the ACPI (Advanced Configuration and Power Interface) to shut down the computer or to go to standby mode. It can happen that while the system is processing a newly installed tool that the screen saver is not activated by the ACPI or that after exiting the tool that Windows XP cannot be shut down properly.

If the TIA Portal is running, the Standby function of the computer is deactivated. To put the computer into Standby, you have to first exit the TIA Portal.

The following description shows several optional settings in the "Power Options Properties" you can use to set the Standby mode of the computer with the function "Hibernate":

1. In Windows XP, open the "Power Options Properties" by pointing at "Start > Settings > Control Panel > Power Options" and select the tab "Hibernate". Select the "Enable hibernation" check box.
2. Switch to the "Advanced" tab. In the "Power buttons" dialog, click the drop down list box under "When I close the lid of my portable computer:" and select the option "Hibernate".
3. Then click the drop-down list box under"When I press the power button on my computer:" and select the option "Shut down".

4. Click "Apply" and confirm the settings with "OK".
5. Afterwards, restart the PC.

If you experience problems shutting down the computer, make sure that the TIA Portal has closed completely.

1. In the shortcut menu, select the Task Manager from the shortcut menu on the Taskbar.
2. If you see the process "Siemens.Automation.ObjectFrame.FileStorage.Server.exe" in the "Processes" tab, wait until this process has closed.
3. Then you can shut down the computer.

Subnet addressing for CP 1613 and CP 1623

CP 1613 and CP 1623 are communication modules with microprocessor. To ensure secure management of communication links, these are processed on the module. The protocol stack in your PC is used for diagnostic purposes (SNMP, DCP). To allow both protocol stacks (i.e. CP 1613/23 Firmware and CP 1613/23 NDIS access) access to the same partners, is recommended to place both stacks of a module in the same subnet.

Editing a device IP address

Do not use the address range from 192.168.x.241 to 192.168.x.250 when editing a device IP address. If necessary, this address range is automatically assigned by the system to a programming device. Depending on the subnet mask, this applies also for all network classes.

"TM_MAIL" instruction for TeleService

The "TM_MAIL" instruction is not included in the scope of delivery for TIA portal V11.0. The information on the use and functionality of the "TM_MAIL" instruction contained in the information system does not apply.

Migrating projects with the TIA Portal

After the migration of hardware configurations and program blocks from earlier automation solutions, first check the functionality of the migrated project before you use it in productive operation.

Screen display

After long periods of work, it can happen in the case of certain computer configurations with Windows XP that parts of the TIA Portal interface are no longer updated. Reducing the graphic hardware acceleration can correct this problem. You can find the setting for this by clicking on the desktop and selecting "Properties > Settings > Advanced > Troubleshoot > Slider Hardware acceleration" by means of the right mouse button. In this dialog, move the slider "Hardware acceleration" gradually to the left and apply this setting until the contents of the screen are displayed correctly again.

Tablet PCs

For the TIA Portal V11, tablet PCs are only released with the operating system Windows 7, not with Windows XP.

SQL Server

During the installation of the SQL Server 2005 or the SQL Server 2005 Express, an error can occur if the product SQL Server 2008 is already installed on your system.

In order to be able to use both varieties of the SQL Server in parallel, the SQL Server 2005 must be installed before the SQL Server 2008. To do this, follow these steps:

1. Remove the SQL Server 2008 version, without deleting the databases.
2. Install WinCC. The SQL Server 2005 is hereby installed and set up on your system.
3. Install the SQL Server 2008 again. Now you can continue to use the databases already created in the SQL Server 2008.

Working with automatically synchronized network drives

Automatic synchronization after a network interruption can result in current (local) project data being stored as a "backup" on the network drive through user interactions. This could cause outdated project data to be loaded from the network drive when opening the project. For this reason, we do not recommend that you store TIA Portal projects on synchronized network drives.

If, however, you do work on synchronized drives, you can continue working locally in the event of a network interruption. In this case, you must always ensure that the TIA Portal application is closed while data is synchronized. The synchronization itself must be implemented in such a way that the current (local) project data replaces the project data on the network drive.

Entry of decimal places

With certain Windows language settings, it may occur that the entry of values with a comma is not recognised (entering "1,23" leads to an error). Instead, use the international format ("1.23").

FAQs on the TIA Portal

FAQs on the TIA Portal are available at http://support.automation.siemens.com (http://support.automation.siemens.com/WW/view/en/28919804/133000).

2.1.2 Notes on the installation

Contents

Information that could not be included in the online help.

Target directory of the installation

Do not use any UNICODE characters (for example, Chinese characters) in the installation path.

Installation of STEP 7 Basic V11 and STEP 7 Professional V11 under Windows XP with Turkish Regional and Language Options

Installation of STEP 7 Basic V11 and STEP 7 Professional V11 under Windows XP may be cancelled, if the regional and language options are set to Turkish. In this case change the regional and language options from Turkish to English or German.

1. Open the Control Panel under Windows with one of the following commands:
   - "Start > Control Panel" (Start menu under Windows XP)
   - "Start > Settings > Control Panel" (classic start menu)
2. Open the "Regional and Language Options".
3. Select the "Regional Options" tab.
4. Under "Standards and formats" select "German" or "English" in the drop-down list.
5. Click "Apply" and confirm with OK.
6. Restart your PC for the setting to become active. Now you can continue with the installation of STEP 7 Basic V11 and STEP 7 Professional V11.
7. After installation, you can revert the regional and language settings (as described in steps 1 to 4) to Turkish.

Removing

In rare cases removal of the program can cause the computer to freeze, even when a full version of SQL Server 2005 is installed. In this occurs, disconnect the computer from the network to continue the removal process.

Installation of the SIMATIC USB driver under Windows server 2003 R2 StdE SP2

An operating system message relating to the SIMATIC USB driver is issued on the operating system Windows Server 2003 R2 StdE SP2. This message must be acknowledged with "Yes" as soon as possible after the message has been issued. The message may be in the background and therefore may not be immediately visible. After a certain period of time, the setup continues with the next component. The SIMATIC USB drivers are then not installed and cannot be used.

Display of the desktop icon

If you do not select the standard installation path during the installation of the TIA portal, the desktop icon may not be displayed correctly. This does not affect the functionality of the product.

2.1.3 Use of communications processors

Contents

Information that could not be included in the online help.

Necessary hotfixes for the use of certain communications processors

For the following communication processors, hotfixes are required for the use of certain products:

- CP 1613
- CP 1613 A2
- CP 1623
- CP 5613
- CP 5614
- CP 5613 A2
- CP 5614 A2
- CP 5623
- CP 5624

Install the following hotfixes and service packs:

- SIMATIC NET CD V8.0 SP1 for Windows 7: Hotfix 1 for SIMATIC NET CD V8.0 SP1
- SIMATIC NET CD 2008 SP3 for Windows XP, Windows Server 2003 or Windows Server 2008

You can find the hotfixes in the Service & Support area (http://support.automation.siemens.com/WW/view/en/12660737) of the Siemens website under the entry ID 12660737

Operation of communications processors

If both STEP7 V5.5 and STEP7 V11 are installed on a computer and STEP7 V5.5 is removed, the setup repair function of STEP7 V11 must be executed before the following communications processors can be used:

- CP 561x
- CP 5613 A2
- CP 5614 A2
- CP 5623
- CP 5624
- CP 5711

2.2 STEP 7 Basic

2.2.1 Notes on use

Contents

Information that could not be included in the online help.

Online operation

The simultaneous online operation of STEP 7 V5.5 or earlier and STEP 7 Basic V11 has not been approved.

Simultaneous online connections on an S7-1200 CPU

It is not possible to establish an online connection from multiple TIA portal instances simultaneously to the same S7-1200 CPU.

Configuring and assigning module parameters

For an overview of the modules you can configure and assign with STEP 7 Basic V11 go to http://support.automation.siemens.com (http://support.automation.siemens.com/WW/view/en/28919804/133000).

Removing/inserting the memory card

After removing or inserting a memory card, always perform a memory reset on the CPU in order to restore the CPU to a functional condition.

Removing and inserting Ethernet modules

If Ethernet modules are removed and re-inserted during operation, you must boot the PC; otherwise, the "Accessible devices" functionality in STEP 7 or NCM PC will not display all devices. While the PC boots, Ethernet modules must be activated.

Notes on the information system

The following function is already described in the information system, but is not available in STEP 7 Basic V11.0 SP1:

- Loading hardware configurations from the target system to the PG/PC.

2.2.2 Editing devices and networks

2.2.2.1 General information on devices and networks

Contents

Currently, there is no general information available on devices and networks.

2.2.2.2 System reaction of the S7-1200 CPU with firmware version V2.0

Contents

Information that could not be included in the online help.

System reaction of the S7-1200 CPU with firmware version V2.0 to events without OB start

When a module of the distributed I/O (PROFINET or PROFIBUS) is removed or inserted, the CPU generates a diagnostic buffer entry and remains in RUN mode.

If a programming error or an I/O access error occurs in a block for which you do not have local error handling activated, then:

- The CPU generates a diagnostic buffer entry.
- Contrary to the description in the online help, the CPU remains in RUN mode.

2.2.2.3 Replacing the ET 200S pulse generator and positioning modules

Contents

Information that could not be included in the online help.

Replacing the ET 200S pulse generator and positioning modules

This information is relevant to the "2 Pulse" pulse generator modules (6ES7 138-4DD00-0AB0) and "1 Step 5V" positioning modules (6ES7 138-4DC00-0AB0) from a project which was created with TIA Portal V11.0. When replacing these modules from the TIA Portal V11.0 with a new version of these modules, the parameter settings are reset to the default values.

This is the case with one of the following procedures:

- Replacting the pulse generator module 6ES7 138-4DD00-0AB0 or the positioning module 6ES7 138-4DC00-0AB0 with their successor modules 6ES7 138-4DD01-0AB0 or 6ES7 138-4DC01-0AB0 by means of a device replacement.
- Updating the module version using the appropriate button in the device properties in the Inspector window.

2.2.2.4 Setting flow control for CM 1241 (RS232)

Contents

Information that could not be included in the online help.

Values for XON and XOFF

If flow control is enabled for the CM 1241 (RS-232) communications module and set to "XON/XOFF", you can enter identical values for the XON and XOFF characters. From a technical point of view, however, this configuration is impractical. You should therefore use different values for XON and XOFF.

2.2.2.5 Notes on online and diagnostics

Contents

Information that could not be included in the online help.

Hardware detection followed by online connection

When the "Online > Hardware detection" command is performed for an unspecified CPU, the online configuration is not loaded from the CPU. If you do not load the configuration resulting from the hardware detection to the CPU, the device and network views will always show a difference between the offline and online configurations. It will appear there are different configurations in the online and diagnostic views, although the MLFBs are identical in the actual CPU and the offline CPU.

Assigning an IP address

If an IP address is assigned directly to a PLC via the diagnostics and online function with "Functions > Assigning an IP address", this IP address will be set permanently and retained even after a restart or power failure.

2.2.2.6 Compiling the hardware of a pulse generator

Contents

Information that could not be included in the online help.

Compiling with a disabled pulse generator

If a pulse generator is deactivated and the following error message nevertheless appears during compilation of the hardware "Pulse generator as: PTO cannot be selected. Associated high speed counter not correctly configured.", follow these steps:

1. Deactivate the high-speed counter.
2. Activate the pulse generator and set the operating mode to "PTO".
3. Deactivate the pulse generator.
4. Recompile the hardware.

2.2.3 Programming a PLC

2.2.3.1 General notes on PLC programming

Contents

Information that could not be included in the online help.

Programming language SCL

For CPUs of the S7-1200 series, SCL is not yet available in STEP 7 V11.0. This functionality will be provided as part of a service pack as soon as possible.

Downloading inconsistent programs to a device

In TIA portal, it is not possible to download inconsistent programs to a device without a consistency check. During the loading process, all blocks of the program are implicitly checked and are compiled again in the event of inconsistencies. If, however, there are programs on your CPU which were loaded with earlier versions of STEP 7, these programs could demonstrate inconsistencies.

In this case, note the following:

If you load an inconsistent program from a device, you will not be able to load the program unchanged to the device afterwards, because a consistency check always takes place during the loading process and existing inconsistencies are corrected.

Copying overlaying tags

Tags that overlay other tags using the keyword "AT" cannot be copied. In this case, copy the overlaid tag and define the overlay again on the copy.

Process image of PTO/PWM outputs

Do not use PTO/PWM outputs in the process image (for example, for accesses in the user program, for online functions or in HMI). The update rate of the process image is much slower than the rate of the signal changes. The display in the process image does not reflect the signal flow.

Conversion of know-how protected blocks from V10.5

After the conversion from V10.5 to V11.0, the program must be compiled. If you use know-how protected blocks, you are prompted to enter the password.

2.2.3.2 Instructions

Contents

Information that could not be included in the online help.

MODBUS library

The instruction "MB_SLAVE" was updated in STEP 7 V10.5 SP2.

If you have already used "MB_SLAVE" V1.0 in a project that was created with STEP 7 V10.5 SP1, you must manually replace this version with the new version MB_SLAVE" V1.1 after the installation of STEP 7 V11.

To do this, follow these steps:

1. Delete "MB_SLAVE" V1.0 from all blocks in the project.
2. Delete "MB_SLAVE" V1.0 from the project library.
3. Insert "MB_SLAVE" V1.1 in all required locations of use.
4. Compile the project.

Using instructions with parameters of type VARIANT in logic blocks with different access types (S7-1200)

Logic blocks (FBs/FCs) and data blocks (DBs) can be created with different access types ("standard" and "optimized"). In logic blocks, you can call any instructions. Certain instructions (for example, "WRIT_DBL" and "READ_DBL") use pointers of type VARIANT at input and output parameters to address data blocks.

Ensure that you do not use these instructions in programs in which logic blocks of different access types are called reciprocally. This could cause the following to occur:

- A structure from a standard data block is directly or indirectly passed to an optimized logic block, which forwards this structure directly or indirectly to one of the blocks mentioned above.
- The reverse scenario, whereby a structure from an optimized logic block is directly or indirectly passed to a standard data block, which forwards this structure directly or indirectly to one of the blocks mentioned above.

2.2.3.3 Testing the user program

Testing with the watch table

Contents

Information that could not be included in the online help.

Multiple access to the same CPU

Access to a CPU from a PG/PC is permitted only when a TIA portal is open. Multiple access to the same CPU is not permitted and can lead to errors.

Modify with trigger

When modifying with a trigger, for example when permanently modifying a tag, an existing control job is aborted if the CPU memory is currently being reset (MRES). The control job is also terminated even if you answer "no" to the prompt asking whether to stop modifying with trigger in the watch table dialog.

Rounding floating-point numbers

In the watch table, floating-point numbers are stored as binary numbers in IEEE format. Since some floating-point numbers (real, long real) that can be displayed in the user interface cannot be mapped exactly to the IEEE format, it is possible that floating-point numbers will be rounded.

If a floating-point number has been rounded for this reason and it is then copied to another input cell in the watch table, the rounding may result in a slight deviation.

Loading data blocks during an active control job

NOTICE
Loading changed data blocks during an active control job can result in unforeseen operating states. The control job continues to control the specified address, although the address allocation may have changed in the data block. Complete active control jobs before loading data blocks.

Testing programs converted from STEP 7 V10.5.

To monitor and test a program converted from STEP 7 V10.5, you have to first compile and load with STEP 7 V11.0.

2.2.4 Technology functions

2.2.4.1 Notes on technology functions (S7-1200)

S7-1200 technology functions

There are no notes available on the S7-1200 technology functions.

2.3 WinCC Readme

2.3.1 Notes on use

Contents

Information that could not be included in the online help.

Parallel installation in the TIA portal

You will be prevented from starting the TIA portal if you set up a non-permitted parallel installation of STEP 7 and WinCC.

The following parallel installations are not permitted in the TIA portal:

- STEP 7 V11 and WinCC V11 SP1
- STEP 7 V11 SP1 and WinCC V11

Both Engineering Systems must always have the same version after an installation.

A dialog opens during installation to inform you of any inconsistencies in your parallel installation.

Copying HMI devices with HMI connections

If you copy an HMI device with HMI connections to a PLC, the HMI connection in the new HMI device will not automatically be connected to an existing PLC with the same name. This statement applies to copying within a project as well as copying across projects.

To access the PLC tag via HMI tag in the new HMI device, you will have to complete the HMI configuration immediately after the copying step. Proceed as follows:

1. Open the "Devices & Networks" editor.
2. Connect the new HMI device with the desired network.
3. Open the connection table.
4. Select the HMI connection of the new HMI device.
5. Select the desired PLC under "Partner".

If you compile the new HMI device or connect additional PLC tags in between copying the HMI device and completing the connection, there may be some instances in which you create an additional HMI connection to the same PLC. This is especially true if you connect HMI tags with DB array elements.

Device replacement

After an HMI device has been replaced, you should check the appearance of the configured screens. Changing the size of the display may result in changes to the position and appearance of screen objects, e.g. recipe view and alarm view.

Device replacement - communication

If an HMI device is replaced, error messages of the type "... will not be supported in the new configuration. It will be removed" will be issued. These alarms refer to configured connections of the device and are triggered, for example, if the HMI devices have a different number of interfaces. These connections are marked red after a device replacement. If you would like to continue to use these connections, you have change the configuration of the connection. Proceed as follows:

1. Open the "Devices and Networks" editor.
2. Click "Network" in the toolbar of the network view.
3. Network the interface of the HMI device with the interface of the CPU.
4. Click in the tabular area of the network view on the "Connections" table.
5. Select the connection marked red.
6. Enter the new interface under "Properties > Properties > General > Interface" in the Inspector window.

Specifying the time of modification in the overview window

The times of modification displayed in the overview window only refer to changes to the object itself. Changes to subordinate objects, e.g. screen objects in a screen, do not cause the time of the last change to the screen to change in the overview window.

HMI device wizard

When you create a device with a color display using the HMI device wizard, the graphics of the navigation buttons may be displayed in black and white. This error only occurs, however, if the new device is created with the same name as a device with a monochrome display which has been deleted in the meantime.

You can avoid this error by always deleting the associated graphics in the Graphics collection whenever you delete a device from the project.

Objects with object references in the project library

Two copying methods can be used in WinCC flexible.

- With "simple copy" a WinCC flexible screen including an IO field, for example is copied. Only the object name of a tag configured on the IO field is copied, as this is a reference.
- With "copy", a screen, an IO field contained there and a tag configured on the IO field together with its properties are copied.

These two methods can also be used for storing an object in a library. Project libraries and the objects contained there are migrated during migration and can be used in WinCC.

In WinCC, however, only one copying method is available. With regard to tags, it functions like "simple copy" in WinCC flexible. With regard to graphics, graphics lists and text lists, it functions like "copy" in WinCC flexible.

If you stored objects with references to tags in a library in WinCC flexible, you must reconfigure the referenced objects when using them in WinCC.

Installing East Asian project languages on a PC without Asian operating system

If you select an East Asian project language on a PC that does not have an Asian operating system installed, the default font will be marked as invalid in the "Runtime settings > Language & font" editor.

To resolve this problem, open the "Regional and Language Options > Languages" dialog in the Control Panel and select the "Install files for East Asian languages" option.

Windows 7 and Windows 7 SP1

WinCC V11 SP1 is released for:

- Windows 7
- Windows 7 SP1

Parallel installation of WinCC V11 SP1 and WinCC V7.0

A parallel installation of WinCC V11 SP1 and WinCC V7 is not released for versions prior to WinCC V7 SP3.

2.3.2 Migration

Contents

Information that could not be included in the online help.

Changing the names of alarm classes

In contrast to WinCC flexible, the names of the predefined alarm classes are not dependent on the user interface language currently in use. During migration, the names of the alarm classes are assigned as follows:

WinCC flexible	WinCC
Error	Errors
System	System
Warnings	Warnings

The display names of the alarm classes can be changed as necessary after migration.

Project languages in WinCC

WinCC V11 does not support all project languages that were available in WinCC flexible, such as Arabic. If you receive an empty project as the result of your migration, you may want to check the set editing language. Do not set the project languages that are not supported as editing language in the source project. Proceed as follows:

1. Open the project with WinCC flexible.
2. Change the editing language to English, for example.
3. Save the project.
4. Restart the migration.

Objects with object references in the project library

Two copying methods can be used in WinCC flexible.

- With "simple copy", a WinCC flexible screen including an IO field, for example, is copied. Only the object name of a tag configured on the IO field is copied, as this is a reference.
- With "copy", a screen, an IO field contained there and a tag configured on the IO field together with its properties are copied.

These two methods can also be used for storing an object in a library. Project libraries and the objects contained there are migrated during migration and can be used in WinCC.

In WinCC, however, only one copying method is available. It functions like "simple copy" in WinCC flexible.

If you have stored objects with references to other objects in a library in WinCC flexible, you must reconfigure the referenced objects when using them in WinCC.

Structures with data type "Stringchar"

In WinCC V11, the data type "StringChar" is not supported in HMI user data types.

If you have used this data type in a structure in a WinCC flexible project, an invalid element is created by the migration in the HMI user data type.

You have to rework this user data type in WinCC. You can only remove the invalid element if you assign a valid data type and release the user data type. You can then delete the element. However, doing this causes any interconnections present between a faceplate and the elements of the user data type to be lost.

Also check the offsets of the subsequent elements and change them, if required.

Tag logging archives from WinCC V7

It is not possible to use the tag logging archives from WinCC V7 after migration to WinCC V11.

2.3.3 Engineering System

2.3.3.1 Screens and Screen Objects

Contents

Information that could not be included in the online help.

Text format of output fields in alarm text

It is not possible to underline tags and text list entries.

Copying display objects between two projects or two devices

In Project_1 configure an alarm window in the Global Screen, for example. You copy the alarm window and paste it in the Global Screen in Project_2.

The enabled alarm classes are partly not enabled in the alarm window after pasting.

This behavior applies to the following display objects:

- Alarm window
- Alarm indicator
- Alarm view

Representation of the cross-references in the Inspector window

The Inspector window displays objects used by a screen object in the "About > Cross-reference" tab.

A screen is open and an object selected. You are using an HMI tag at the object as process tag.

The object and the linked HMI tag are displayed in the cross-references. All locations of use of the object and the HMI tags are listed.

If the HMI tag is interconnected with a PLC tag or a DB tag, then the locations of use of the interconnected PLC tag or DB tag will be displayed.

Event names in case of alarms in the "Info" tab of the Inspector window

In some alarms of the Inspector window the event names in the "Info" tab will deviate from the names in the "Properties" tab.

Name in the "Properties" tab of the Inspector window	Name in the "Info" tab of the Inspector window
Cleared	ClearScreen
Loaded	GenerateScreen
Enable	Activate
Change	Change
When a dialog is opened	ONMODALBEGIN
When a dialog is closed	ONMODALEND
User change	PASSWORD
Screen change	SCREEN
Disable	Deactivate
Press	Press
Outgoing	Going
Incoming	Coming
Limit "high limit error" violated	AboveUpperLimit
Limit "low limit error" violated	BelowLowerLimit
Click	Click
Loop-In-Alarm	LoopInAlarm
Release	Release
Alarm buffer overflow	OVERFLOW
Acknowledge	Acknowledgement
Runtime stop	Shutdown
Press key	KeyDown
Release key	KeyUp
Switch ON	SwitchOn
Switch OFF	SwitchOff
Value change	Change value

Dynamization of object properties in a group

The dynamization of properties for all objects of the group which have these properties is not possible in a group. In WinCC V11, the properties of the objects beloning to a group can only be dynamized for each object itself.

Illegible characters in Runtime Professional

With Runtime Professional, only characters belonging to the language area which is defined with the operating system setting "Language for non-Unicode programs" can be displayed on the target system. Texts with characters from other language areas can, however, also be configured in the project.

Illegible characters may occur in the Engineering System with the objects text field, symbolic I/O field, gauge and slider if the settings in the operating system relating to "Language for non-Unicode programs" do not match the selected editing language and the objects are displayed in a different design than "WinCC Classic". The characters are displayed correctly in the Inspector window and the "Project texts" editor.

Therefore, first check in the Control Panel under "Regional and Language Options > Advanced" whether the setting for "Language for non-Unicode programs" is the same as the editing language. Otherwise, you can check or change the correct texts in the Inspector window or the "Project texts" editor.

Faceplates

Faceplates cannot be rotated or mirrored.

Persistence with display objects in WinCC Runtime Professional

The objects f(t)-trend view, f(x)-trend view, alarm view, recipe view, table view and value table have settings for the persistence of online configurations. If you have configured "Persistence" for "Online configuration" and "Keep changes" for "Reaction to screen change", you can make changes to the configuration dialogs in runtime which will be retained after a screen change and after runtime is exited.

However, online configurations for the settings mentioned cause changes to the configuration of the objects in the Engineering System to only be applied in Runtime if you recompile the device with the command "Compile > Software (rebuild all)".

Basic Panels, OP73, OP77A and TP177A: Displaying texts in runtime

The default font selected in the "Runtime settings > Languages & font" editor has an effect on the display of texts in runtime.

Text entries may be truncated if you selected an unfavorable font size or style.

This setting possibly has an effect on the following text entries:

- Tooltips
- long alarm text
- text in the dialogs

2.3.3.2 Tags

Contents

Information that could not be included in the online help.

Tag names

HMI tag names may not start with the character @.

Display of deleted array elements at location of use of HMI tags

The locations of use of HMI tags, such as the process value of IO fields, are usually indicated by the tag name. If the element of an array tag is used, then the tag name will be extended by the index of the array element indicated in brackets.

If a used tag is no longer included in the project, then the tag name will still be displayed at the location of use. The field will be displayed with a red background to indicate the missing tag. If a used array element or the array tag is no longer present, then only the index of the array element will be displayed in brackets. The tag name will not be displayed. The field is highlighted in red. You can no longer identify the name of the associated array tag based on the location of use in this instance.

If you do not know which array tag was linked to this location of use, then it may be necessary to link the array element once again.

If a tag or array tag was created based on a reference, then the selected reference will be closed automatically.

If an HMI tag is connected with an array element of a PLC tag and the PLC tag does no longer exist in the project, then the same behavior will take place in the "HMI tags" editor.

Array tags as list entry of multiplex tags

You can use the array tags of the Char data type just like the tags of the String data type.

The use of an array tag of the Char data type as list entry of a multiplex tag in the "HMI tags" editor is not supported.

2.3.3.3 Alarm system and alarm displays

Contents

Information that could not be included in the online help.

Displaying special characters in alarm texts

When configuring alarm texts, a fixed character set is used in the Engineering System. This character set allows you to use numerous special characters in alarm texts.

Language-dependent fonts are used in runtime to display the texts, for example MS PGothic, SimSun. The fonts used in runtime do not support all special characters. As a result, some special characters are not displayed in runtime.

Use of multiplex tags in output boxes with alarm texts

In the engineering system, it is also possible to use multiplex tags in the output boxes of alarm texts. During runtime, this leads to an incorrect display of the alarm, because the use of multiplex tags is not supported by the basic panels.

2.3.3.4 System functions

Contents

Information that could not be included in the online help.

2.3.3.5 Recipes

Contents

Information that could not be included in the online help.

2.3.3.6 User administration

Contents

Information that could not be included in the online help.

2.3.3.7 Communication

Contents

Information that could not be included in the online help.

Connection interruptions with Mitsubishi PLCs

After multiple connection interruptions, a situation may arise where all the connection resources of the Mitsubishi PLC are in use and the connection can no longer be established. It is recommended to check these connection resources in the PLC program of the PLC and also enable them again.

Using "DTL" data type for area pointers

Use the "DTL" data type for configuration of area pointers "Date/time" and "Date/time PLC". The "DTL" data type supports time information down to the nanosecond range. Because Basic Panels support time information only down to the millisecond range, you will encounter the following restrictions in the case of use on the area pointers:

- Area pointer "Date/time"
  For transmission of time information from a Basic Panel to the PLC, the smallest unit of time is 1 millisecond. The value range from microseconds to nanoseconds of the "DTL" data type will be filled with zeros.
- Area pointer "Date/time PLC"
  For transmission of time information from a PLC to a Basic Panel, the area from microseconds to nanoseconds will be ignored. The time information will be processed on the panel down to milliseconds.

Limited number of possible HMI connections

An error message is displayed during compilation of a device indicating that the configuration of the HMI connection in the "Devices & Networks" editor is invalid. The reason may be that the maximum number of possible connections of the HMI device or PLC has been exceeded.

Check the maximum number of available connections. Consult the device manuals of the devices you are using.

Routed communication with S7 300/400

The communication of connection partners in various subnets can be routed via the following links: PROFINET, PROFIBUS, MPI.

Using PROFINET IO with panel HMI devices

When using PROFINET IO to connect the direct keys and LEDs of HMI devices to the PLC, you can define an offset for the address area of the inputs and outputs during configuration in HW Config.

The following restriction applies when you use a PROFINET IO-capable CPU of the 400 series with one of the HMI devices listed below:

The offset for the start of the address area of the inputs must not be bigger than the offset for the start of the address area of the outputs.

The restriction applies to the following HMI devices:

- OP 177B
- OP 277
- Mobile Panel 177

For the configuration of the address parameters, open the PLC with the CPU of the 400 series in HW Config. Select the HMI device connected via PROFINET IO in the station window of HW Config. A table with the properties of the HMI device is displayed at the bottom of the station window in the detail view. Select the line containing the addresses of the HMI device in the table and open the object properties using the shortcut menu.

Select the "Addresses" tab in the "Object properties" dialog. Configure the offset for the inputs under "Inputs > Start". Configure the offset for the outputs under "Outputs > Start".

Exceeding value ranges with Mitsubishi MC and Mitsubishi FX

With some data types, the communication drivers Mitsubishi MC and Mitsubishi FX do not check whether the value of a recipe tag exceeds the value range of the PLC tags. The data types affected are:

- 4-bit block
- 12-bit block
- 20-bit block
- 24-bit block
- 28-bit block

Area pointer Coordination in an OPC connection

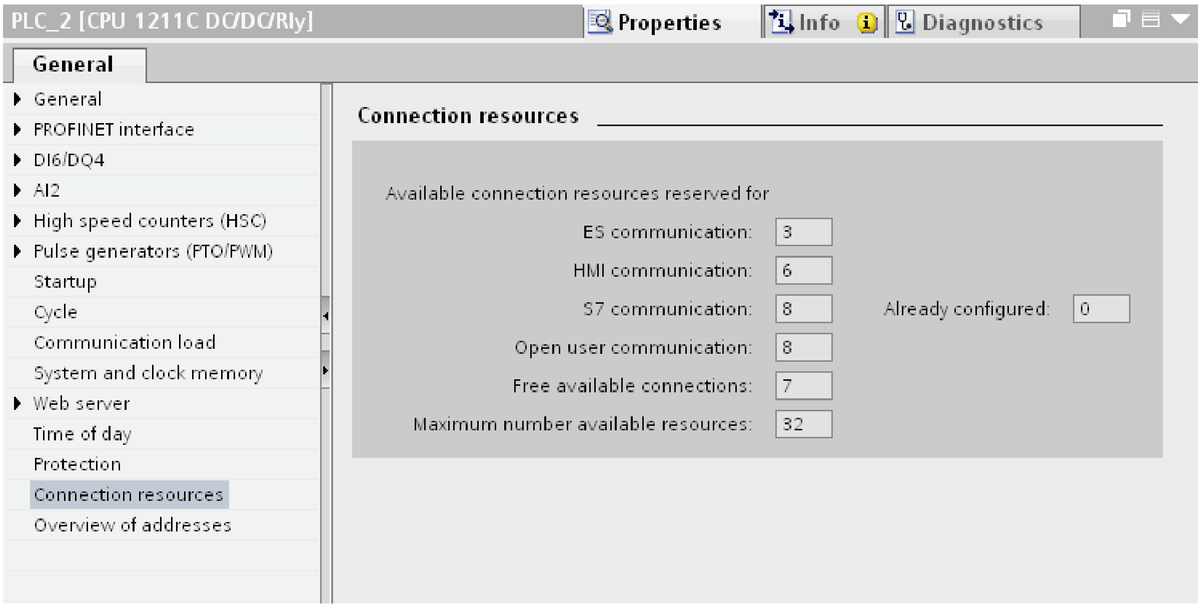
In principle, the area pointer Coordination can be used eight times in an OPC connection. If you have configured an OPC connection and automatically create another OPC connection using "Add", the area pointer Coordination is only displayed once in the newly created connection. In this case, you should change the communication driver of the connection. If you then set OPC again as the communication driver, the area pointer Coordination can again be used eight times.

Communication resources: SIMATIC S7 1200

The SIMATIC S7 1200 controller provides six communication resources for HMI communication.

The number of HMI connections that you can actually configure depends on the HMI devices that you connect with the SIMATIC S7 1200.

One HMI Panel occupies one communication resource per connection.



Connection resources

Available connection resources reserved for

ES communication: 3

HMI communication: 6

S7 communication: 8 Already configured: 0

Open user communication: 8

Free available connections: 7

Maximum number available resources: 32

Valid operand types for communication with Mitsubishi MC TCP/IP

Operand type "W" is not available for the "String" data type.

2.3.4 Compiling and loading

Contents

Information that could not be included in the online help.

Compiling and loading

If internal errors or warnings occur during compiling, compile the complete project using the command "Compile > Software (rebuild all)" in the shortcut menu of the HMI device.

Before you start productive operation with your project, compile the entire project using the "Compile > Software (rebuild all)" command from the shortcut menu of the HMI device.

If you are using HMI tags that are connected to the control tags in your project, compile all modified blocks with the command "Compile > Software" in the shortcut menu before you compile the HMI device.

Saving the WinCC project

If you save a project in WinCC using the "Save As…" command, this has no effect on the name of the Runtime projects generated for the devices. If you do not adapt the target path of the devices in the "Extended download to device" dialog, the Runtime projects on the target devices will be overwritten.

Settings for update of operating system

If you select the command "Online > HMI device maintenance > Update operating system" from WinCC, you cannot change the settings such as the type of PG/PC interface or baud rate. The settings used during the last download are always used.

To make changes to the settings, open the "Extended download" dialog using the "Online > Extended download to device" command and change the settings. When you click the "Load" button the changed settings are saved.

Alternatively, you can perform an update of the operating system with changed settings with ProSave. You start ProSave via the Windows Start menu "Siemens Automation > Options and Tools > HMI Tools > SIMATIC ProSave".

Incorrect installation of ProSave

If you receive an error message during installation of ProSave when loading data to a target device or maintenance of an HMI device, then you cannot remedy this error using the repair function of setup. Remove ProSave via the Control Panel. Then start setup and install the "ProSave" component again.

Checking the address parameters

During compilation of an HMI device in the project tree with the command "Compile > Software" in the shortcut menu, the address parameters of the HMI device, such as the IP address, will not be checked. If you want to ensure that the address parameters are checked as well, you will have to compile the HMI device in the "Devices & Networks" editor of the toolbar.

Error message when downloading data to the PLC

A panel and a PLC are connected and communicating with other.

If a tag is accessed while downloading data from the panel to the PLC, an error message is displayed on the panel.

Delayed reaction in the "Extended download to device" dialog

If the settings in the "Extended download to device" dialog for "Type of the PG/PC interface" and "PG/PC interface" do not match the settings on the HMI device, this can result in the application not responding for up to a minute.

Extended download with an S7-1200 and a Comfort Panel

An S7-1200 PLC and a Comfort Panel are located in the same physical network as the PG/PC. You open the "Extended download to device" dialog for the Comfort Panel.

If you activate the option "Show all accessible devices", it may occur that the application stops responding.

OP77A, OP73, TP177A: Loading projects

When loading a project to an HMI device, it can happen that Runtime is not automatically ended, even though "Remote Transfers" is activated in the Panel.

If this happens, stop Runtime and manually set the transfer mode on the HMI device.

Loading a SIMATIC HMI application to a PC station

The following circumstances can lead to an error message during the first load of a SIMATIC PC station:

- A SIMATIC HMI application is configured in a PC station in the project
  - WinCC Runtime Advanced
  - WinCC Runtime Professional
  - WinCC Standby
  - or WinCC Client
- The property "S7RTM is installed" is activated.

Before you load a SIMATIC PC station for the first time, select the configured device HMI_RT (WinCC...) in the project navigation. Open the "Extended download to device" dialog and select the appropriate interface and parameter settings. Click "Load".

You then load the PC station as normal.

Unpacking a distributed "Pack&Go" file

To unpack a "Pack&Go" file which is distributed over several files, you need an unpack program. The unpack program integrated in the operating system is not adequate to unpack distributed files.

2.3.5 Runtime

2.3.5.1 Notes on operation in Runtime

Contents

Information that could not be included in the online help.

> ⚠ CAUTION
>
> **Ethernet communication**
>
> In Ethernet-based communication, the end user is responsible for the security of his data network. The proper functioning of the device cannot be guaranteed in all circumstances; targeted attacks, for example, can lead to overload of the device.

Special characters in the user view

Special characters, such as / " § $ % & ' ?, are not permitted when entering a name or the password in the user view.

Language behavior - Layout of screen keyboard

The layout of the screen keyboard is not switched when the runtime language changes.

Tag values overwrite the maximum length

You enter a character string in a string tag via an I/O field. If the character string exceeds the number of configured tags, the character string will be shortened to the configured length.

2.3.5.2 Notes on operation of panels in Runtime

Contents

Information that could not be included in the online help.

Using the mouse wheel in Runtime

The use of the mouse wheel in Runtime is not supported on all panels.

2.3.6 HMI devices

2.3.6.1 General notes

Contents

Information that could not be included in the online help.

If the PC goes into standby or hibernate mode while the transfer is in progress, the panel status after interruption of the transfer is not defined.

TS Adapter with Ethernet interface

If an HMI device is connected via Ethernet and a TS adapter, it can not be reset to factory settings.

Simulation of the Basic Panels

Use an output field in an alarm text to output an external tag. The content of the output field will always be displayed with "0" during simulation.

Simulation with real PLC connection

The access point used by the simulation is independent from the settings of the Engineering System and can only be altered in the Control Panel with the "Setting PG/PC Interface" tool. If the PLC connection is terminated right after the start of the simulation with alarm 140001, you should check the access point used by the simulation with "Setting PG/PC Interface".

1. Double-click "Setting PG/PC Interface" in the Control Panel. A dialog opens.
2. Select" "S7ONLINE" in the "Access point of application" field as standard for HMI.
3. Select the interface in the "Interface Parameter Assignment Used" area.
4. Exit the dialog "Setting PG/PC Interface" with OK

Loading of projects without recipe data records

You are using recipes in a project. You transfer the project to a Basic Panel without recipe data records.

You may encounter inconsistencies if you have altered the structure of the recipe in the Engineering System and the device already held recipe data records.

Check the consistency of the data records in this case. The device will not issue a note for all structural changes.

Floating point numbers on MP 277, MP 377, TP 177B 4" and CP4

Only floating point numbers in the range from 10^{-293} ... 10^{+307} are displayed correctly on the HMI devices MP 277, MP 377, TP 177B 4" and CP4. If the tag value is outside this range, it is displayed as 0.

USB device driver under Windows XP

If a configuration PC with Windows XP and a Comfort Panel are connected via USB, the S7-USB driver may be reinstalled when the HMI device is restarted. The device settings may in this case not be restored.

Installation 3

3.1 System requirements for installation

3.1.1 Notes on the system requirements

System requirements for individual products

The system requirements may differ depending on the products you want to install. You should therefore check the individual system requirements of your products.

If you want to install several products, make sure that your system meets the demands of the product with the highest requirements.

Displaying PDF files

To be able to read the supplied PDF files, you require a PDF reader that is compatible with PDF 1.7 e.g. Adobe (R) Reader version 9.

Displaying the Welcome Tour

You require the Adobe (R) Reader as of version 9 to start the Welcome Tour for the TIA portal.

See also

Licenses (Page 71)

Starting installation (Page 73)

Displaying the installed software (Page 77)

Modifying or updating installed products (Page 78)

Repairing installed products (Page 80)

Starting to uninstall (Page 82)

3.1.2 System requirements STEP 7 Basic

3.1.2.1 Software and hardware requirements STEP 7

System requirements for installation

The following table shows the minimum software and hardware requirements for installation of the "SIMATIC STEP 7 Basic" software package:

Hardware/software	Requirement
Processor	Pentium 4, 1.7 GHz or similar
RAM	1 GB
Free hard disk space	2 GB on system drive "C:"
Operating systems *	• Windows XP (Home SP3, Professional SP3) • Windows 7 (Home Premium, Professional, Enterprise, Ultimate)
Graphics card	32 MB RAM 24-bit color depth
Screen resolution	1024 x 768
Network	10Mbit/s Ethernet or faster
Optical drive	DVD-ROM

* For more detailed information on operating systems, refer to the help on Microsoft Windows or the Microsoft homepage.

Recommended hardware

The following table shows the recommended hardware for the operation of STEP 7.

Hardware	Requirement
Computer	SIMATIC FIELD PG M2 PREMIUM (or similar PC)
Processor	2.2 GHZ CORE 2 DUO (T7500)
RAM	1X2GB DDR2 RAM
Hard disk	250GB S-ATA HDD
Monitor	15" SXGA+ DISPLAY (1400 X 1050)
Optical drive	DL MULTISTANDARD DVD RW

3.1.3 System requirement for WinCC Basic

3.1.3.1 Software and hardware requirements

Introduction

Specific requirements for the operating system and software configuration must be met for the installation.

Note

WinCC is generally authorized for use in a domain or workgroup.

However, be aware that domain group policies and restrictions of the domain may hinder the installation. If this happens, remove the computer from the from the domain prior to installing Microsoft Message Queuing, Microsoft SQL Server 2005 and WinCC. Log onto the computer in question locally with administrative rights. Then perform the installation. After successful installation, you can enter the WinCC computer back into the domain. If the domain group policies and restrictions of the domain do not impede the installation, the computer need not be removed from the domain during the installation.

Be aware that domain group policies and restrictions of the domain may also hinder operation. If you cannot avoid these restrictions, run the WinCC computer in a workgroup.

Consult with the domain administrator if needed.

Installation requirements

The following table shows the minimum software and hardware requirements that have to be met for installation of the "SIMATIC WinCC Professional" software package:

Hardware/software	Requirement
Processor type	Pentium M, 1.6 GHz or similar
RAM	2 GB
Free hard disk space	2 GB on system drive "C:"
Operating systems *	• Windows XP Professional SP3 • Windows 7 (Professional, Enterprise, Ultimate) • Windows 7 SP1 (Professional, Enterprise, Ultimate) • Windows Server 2003 R2 Standard Edition SP2 • Windows Server 2008 Standard Edition SP2
Graphics card	32 MB RAM 24-bit color depth
Screen resolution	1024x768
Network	Ethernet 10 Mbps or faster

Hardware/software	Requirement
Optical drive	DVD-ROM
Software	Microsoft .Net Framework 3.5 SP1 Microsoft Windows Message Queuing

* For more detailed information on operating systems, refer to the help on Microsoft Windows or the Microsoft homepage.

Simultaneously opening multiple instances of WinCC on a configuration PC can also increase the hardware capacity required.

Note

"Aero Glass Style" of Microsoft Windows 7

A powerful graphics card is required for "Aero Glass Style". It requires DirectX9 capabilities and 128 MB of dedicated graphics memory.

The performance of the architecture of the graphics system can significantly influence the performance of WinCC.

Recommended hardware

The following table shows the recommended hardware for the operation of SIMATIC WinCC.

Hardware	Requirement
Computer	SIMATIC FIELD PG M2 PREMIUM
Processor	2.2 GHZ CORE 2 DUO (T7500)
RAM	1X2GB DDR2 RAM
Hard disk	250GB S-ATA HDD
Monitor	15" SXGA+ DISPLAY (1400 X 1050)
Optical drive	DL MULTISTANDARD DVD RW

Parallel installation of WinCC V11 and other SIMATIC products

The parallel installation of WinCC V11 is permitted with the following other SIMATIC products

- WinCC Basic V11, WinCC Comfort V11 or WinCC Advanced V11 can be installed on a computer parallel to STEP 7 V5.4 or V5.5, STEP 7 Micro/WIN, STEP 7 10.5, WinCC flexible (as of 2008) and WinCC (as of V7.0 SP2).
  When WinCC V7 is removed, the Setup for WinCC V11 should be run and a repair performed for the installation.
- WinCC Professional V11 can be installed on a computer parallel to STEP 7 V5.4 or V5.5, STEP 7 Micro/WIN, STEP 7 10.5 and WinCC flexible (as of 2008).

Installing Microsoft .Net Framework

.Net Framework 3.5 SP1 is included on the installation medium. The installation routine determines if .Net Framework is already installed at the beginning of the installation. If .Net Framework is not installed, you will be prompted with a dialog to perform the installation. When you confirm the prompt for installation, .Net Framework is installed. You need to reboot the computer after installing .Net Framework. If you do not install .Net Framework, the installation of WinCC Runtime Professional is aborted.

Installing Microsoft Windows Message Queuing in Windows XP

You can install the Windows Message Queuing component from the Windows Control Panel.

Click Start > Control Panel. Double-click "Add or remove software", the "Add or remove software" dialog opens. In the "Add or Remove Programs" dialog, click "Add or Remove Windows Components". The Windows Components Wizard opens. Select the "Message Queuing" component in the Windows Components Wizard. Click "Next", the "Message Queuing" component is installed.

Installing Microsoft Windows Message Queuing in Windows 7

You can install the Windows Message Queuing component from the Windows Control Panel.

Click Start > Control Panel. Click "Programs" and the "Programs" dialog opens. Under the "Programs and Features" section, click "Turn Windows features on or off". The "Windows Features" dialog opens. In the "Windows Features" dialog, select the "Microsoft Message Queue Server" feature. Click "OK" to enable the "Microsoft Message Queue Server".

Online help for Windows 7 / Windows Server 2008

Windows 7 and Windows Server 2008 no longer support all online help formats by default. With WinCC, these online help formats are used in the following cases:

- Calling WinCC Direct Help
- Calling the WinCC Information System from the WinCC editors or via the Direct Help links

To be able to continue calling WinCC Direct Help, the following components are installed during the installation of WinCC:

- Microsoft Help Engine

You can also call the WinCC Information System under Windows 7 and Windows Server 2008 from the Windows Start menu or from the installation folder.

To call the WinCC Information System from the WinCC editors or via the Direct Help links, some changes have to be made to the operating system. You can find more information on this in the section "More information for advanced users" of Microsoft Support article "917607": http://support.microsoft.com/kb/917607 (http://support.microsoft.com/kb/917607)

See also

Licensing of WinCC Engineering System (Page 66)

3.1.3.2 Licenses and Powerpacks

Licensing of WinCC Engineering System

You require a license key for the following:

- WinCC Engineering System, for example, WinCC Professional
- Add-ons for WinCC Engineering System

You can install the license key during the installation of WinCC. You transfer the licenses for WinCC add-ons after installation with the Automation License Manager.

Starting without a valid license key

If you start WinCC without a valid license, the system alerts you that you are working in non-licensed mode. You have the option of activating a one-time trial license. Trial licenses for Engineering editions WinCC Basic, Comfort, Advanced und Professional expire after 21 days.

When the trial license expires, the following scenarios can occur:

- WinCC was never licensed on the PC in question.
  - Operations requiring a license can no longer be performed in WinCC.
- WinCC was already licensed on the PC in question.
  - An alert for non-licensed mode is presented every 10 minutes and for every action requiring a license by a window requiring acknowledgment.

License requirements for simulation

When you want to start simulation in WinCC using the menu command "Online > Simulation> With tag simulator", you do not need licenses for WinCC Runtime or licensed-based add-ons.

If the following conditions are met, you also need the appropriate licenses for the simulation of WinCC Runtime and license-based add-ons:

- The engineering station is connected to a PLC.
- The connection to the PLC is configured and active.

You start the simulation with the "Online > Simulation > Start" menu command.

See also

Software and hardware requirements (Page 63)

Licensing of HMI devices (Page 67)

Working with license keys (Page 68)

Licensing of HMI devices

Non-PC-based HMI devices are always equipped to maximum capacity. A license key is not required for runtime operation.

A license is required for each add-on for non-PC-based HMI devices. The license key of the respective license always activates one instance for use.

License key

To be able to license non-PC-based HMI devices with license keys, you require the "SIMATIC HMI License Manager Panel Plug-in" add-on.

WinCC Setup installs this add-on by default. You can open the License Manager Panel plug-in in the Automation License Manager with the menu command "Edit > Connect Target System > Connect HMI Device".

If WinCC is not installed, an installation of ProSave 7.2 or higher is required.

Note

Further information about handling the licenses can be found in the Automation License Manager help.

Note

Verify that the current release of the operating system is installed on the HMI device before you start licensing. If necessary, update the operating system using ProSave.

Data backup

CAUTION

Destruction of license keys on non-PC-based HMI devices

Installed license keys and authorizations are destroyed by the backup/restore processes on the HMI devices listed below.

- 270 series
- 370 series

Carry out the following before beginning restoring:

- Use the Automation License Manager and ProSave to check whether license keys are installed on the HMI device.
- Remove any license keys present on the HMI device.
  After restoring has been carried out, re-install the license keys on the HMI device.

Non-licensed mode

Runtime add-ons can also be used without a license without restriction. An alert for non-licensed mode is presented every 10 minutes by a window requiring acknowledgment.

See also

Licensing of WinCC Engineering System (Page 66)

Working with license keys

Introduction

Install a license key in the following situations:

- To use the WinCC Engineering System
- To use add-ons for the WinCC Engineering System
- To operate WinCC Runtime
- To use add-ons for WinCC Runtime on PC-based HMI devices
- To use add-ons on non-PC-based HMI Devices

To uninstall a license key in the following cases:

- When backing up data
- If you no longer require the license

You can then use this license on another PC or HMI device.

When you install a license, the associated license key is removed from the license key storage location.

Note

A license key cannot be copied. The copy protection employed prevents the license keys from being copied.

Data backup

Remove the license keys on the HMI device when backing up data on the HMI device and when creating a backup during device replacement.

You use the Automation License Manager to back up license keys of a HMI device to the storage area of the license key.

CAUTION

Destruction of license keys on non-PC-based HMI devices

Installed license keys are destroyed by backup/restore processes on the HMI devices listed below.

- 270 series
- 370 series

Carry out the following before beginning restoring:

- Use the Automation License Manager and ProSave to check whether license keys are on the HMI device.
- Remove any license keys present on the HMI device.
  After restoring has been carried out, re-install the license keys on the HMI device.

CAUTION

Destruction of license keys on PCs

Start by removing all license keys in the following situations:

- Before you format the hard disk
- Before you compress the hard disk
- Before you restore the hard disk
- Starting an optimization program that moves fixed blocks
- Installing a new operating system

Read the description of Automation License Manager ("Start > Simatic Automation > Documentation"). Observe all warnings and notices.

The license key storage location on PC-based HMI devices and on non-PC-based HMI devices where Automation License Manager is used may contain multiple license keys. This capability means you can store multiple licenses of the same type at one location. Save all license keys of the HMI device to the same storage location.

CAUTION

Always keep the original storage location of the license keys.

Invalid license after time zone change

The installed license no longer functions in the following case.

- If you change the time zone on a WinCC PC as follows:
  - From a time based on a complete hour to a time not based on a complete hour. Example: You change the time zone from GMT +3:00 to GMT +3:30.

To avoid this inconvenience, uninstall the license key under the time zone setting that was set when the license key was installed.

Example:

You have installed the license key with a time zone setting based on a full hour. Then also uninstall the license key with a time zone setting based on a full hour.

This behavior does not apply to the trial license.

Defective license

A license is defective in the following cases:

- If the license key is no longer accessible at the storage area.
- If the license key disappears during its transfer to the destination drive.

You can use the Automation License Manager to repair the defective license. Use the "Restore" function or the "Restore Wizard" of the Automation License Manager for this purpose. Contact Customer Support in order to restore the license. For additional information see: http://support.automation.siemens.com (http://support.automation.siemens.com)

Note

The runtime software can also be operated without errors if the license is missing or defective. The system alerts you at brief intervals that you are working in non-licensed mode.

CAUTION

If you start WinCC Engineering System without a valid license key, the system alerts you that you are working in non-licensed mode. You have the one-time option of activating a trial license. The trial license expires after 21 days.

When the trial license expires, the following scenarios can occur:

- WinCC was never licensed on the PC in question.

  WinCC can no longer be started.
- WinCC was already licensed on the PC in question.

  WinCC cannot be started. An alert for non-licensed mode is presented every 10 minutes by a window requiring acknowledgment.

See also

Licensing of WinCC Engineering System (Page 66)

3.2 Licenses

Availability of licenses

The licenses for the products of the TIA Portal are usually supplied on the installation data medium and installed automatically by the Automation Licence Manager during the installation process of the TIA Portal.

If you remove the TIA Portal, the corresponding licenses are also removed automatically. Licenses still required should be secured.

Provision of the Automation License Manager

The Automation License Manager is supplied on the installation data medium and is transferred automatically during the installation process.

If you remove the TIA Portal, the Automation License Manager remains installed on your system.

Working with the Automation License Manager

The Automation License Manager is a product of Siemens AG, which is used for handling license keys (technical representatives of licenses).

Software products that require license keys for operation, such as the TIA-Portal, register the required license key automatically with the Automation License Manager . If the Automation License Manager finds a valid license key for this software, the software can be used according to the license usage terms associated with this license key.

Note

For additional information on how to manage your licenses with the Automation License Manager , refer to the documentation supplied with the Automation License Manager .

See also

Notes on the system requirements (Page 61)

Starting installation (Page 73)

Displaying the installed software (Page 77)

Modifying or updating installed products (Page 78)

Repairing installed products (Page 80)

Starting to uninstall (Page 82)

Installation log (Page 72)

3.3 Installation log

Function of the installation log

The progress during the following installation processes is logged in a file:

- Installing products
- Modifying or updating already installed products
- Repairing an existing installation
- Uninstalling products

If errors occur during the installation process or warnings are issued, these can be evaluated with the help of the log file. You can do this yourself or contact product support.

Installation logs storage location

The log file is the most recent file with the file extension ".log" and whose name begins with "SIA", e.g. "SIA_STEP7_PRO_V11.log".

The location of the log file is stored in the environment variable "%autinstlog%". You can enter this environment variable in the address bar of Windows Explorer to open the folder with the log files. Alternatively, you can navigate to the corresponding directory by entering "CD %autinstlog%" in the command line.

The location depends on the operating system, e.g. "C:\Documents and Settings\All Users\Application Data\Siemens\Automation\Logfiles\Setup" in the English version of Windows XP.

Setup_Report (CAB file)

To make it easier to provide Product Support with all necessary files, an archive file that contains the installation log and all other required files is saved in CAB format. This archive can be found at "%autinstlog%\Reports\Setup_report.cab". Send this CAB file to Product Support if you need assistance with installation. With this information, Product Support can determine whether the installation was executed properly. CAB files that were generated during earlier installation processes are saved with a date ID in the "Reports" directory.

See also

Licenses (Page 71)

Starting installation (Page 73)

Installing Support Packages (Page 76)

Displaying the installed software (Page 77)

Modifying or updating installed products (Page 78)

Repairing installed products (Page 80)

Starting to uninstall (Page 82)

3.4 Starting installation

Introduction

Software packages are installed automatically by the setup program. The setup program starts once the installation medium has been inserted in the drive.

Requirement

- Hardware and software of the programming device or PC meet the system requirements.
- You have administrator privileges on your computer.
- All running programs are closed.

Procedure

To install the software packages, follow these steps:

1. Insert the installation medium in the relevant drive.

   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.

2. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.

   The dialog for selecting the setup language opens.

3. Choose the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click the "Read Notes" or "Installation Notes" button.

   The help file containing the notes opens.

5. Once you have read the notes, close the help file and click the "Next" button.

   The dialog for selecting the product languages opens.

6. Select the languages for the product user interface, and click the "Next" button.

   Note

   "English" is always installed as the basic product language.

   The dialog for selecting the product configuration opens.

7. Select the products you want to install:
   – If you wish to install the program in a minimal configuration, click on the "Minimal" button.

   – If you wish to install the program in a typical configuration, click on the "Typical" button.

   – If you wish to personally select the products to be installed, click on the "User-defined" button. Then select the check boxes for the products you wish to install.

8. If you want to create a shortcut on the desktop, select the "Create desktop shortcut" check box.
9. Click the "Browse" button if you want to change the target directory for the installation. Note that the length of the installation path must not exceed 89 characters.
10. Click the "Next" button.

    The dialog for the license terms opens.
11. To continue the installation, read and accept all license agreements and click "Next".

    If changes to the security and permissions settings are required in order to install the TIA Portal, the security settings dialog opens.
12. To continue the installation, accept the changes to the security and permissions settings, and click the "Next" button.

    The next dialog displays an overview of the installation settings.
13. Check the selected installation settings. If you want to make any changes, click the "Back" button until you reach the point in the dialog where you want to make changes. Once you have completed the desired changes, return to the overview by clicking on "Next".
14. Click the "Install" button.

    Installation is started.

    Note

    If no license key is found during installation, you have the chance to transfer it to your PC. If you skip the license transfer, you can register it later with the Automation License Manager.

    If the installation was successful, a message to this effect is displayed on the screen. If errors occurred during installation, an error message is displayed informing you of the type of errors.

    ---
15. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".
16. If the computer does not reboot, click "Exit".

Result

The TIA Portal along with the products and licenses you have ordered and the Automation License Manager have been installed on your computer.

See also

Installation log (Page 72)

Notes on the system requirements (Page 61)

Licenses (Page 71)

Displaying the installed software (Page 77)

Modifying or updating installed products (Page 78)

Repairing installed products (Page 80)

Starting to uninstall (Page 82)

3.5 Installing Support Packages

You can install subsequent support packages, for example, hardware support packages, in the TIA Portal.

Note

Support packages for STEP7 V5.4 or V5.5 cannot be used.

Procedure

To install a Support Package, follow these steps:

1. Click "Support packages" in the "Options" menu.

   The "Detailed information" dialog opens. A table lists all support packages from the directory that you selected as the storage location for support packages in the settings.

2. If you want to install a support package that is not in the list, you have the following options:

   – If the support package is already on your computer, you can add it to the list by selecting "Add from the file system".

   – If you add a support package from the "Service & Support" page on the Internet, first you download it by selecting "Download from the Internet". Then you can add it from the file system.

3. Select the support package that you want to install.
4. Click "Install."
5. Close and then restart the TIA Portal.

See also

Installation log (Page 72)

3.6 Displaying the installed software

You can find out which software is installed at any time. In addition, you can request additional information on the installed automation software to be displayed.

Procedure

To display an overview of the software installed, follow these steps:

1. Click "Installed software" in the "Help" menu.

   The "Installed software" dialog opens. You will see the installed software products in the dialog. Expand the entries to see which version is installed in each case.

2. If you would like to display additional information on the installed automation software, click the link on the "Detailed information about installed software" dialog.

   The "Detailed information" dialog opens.

3. Chose the topic you want more information about in the area navigation.

See also

Notes on the system requirements (Page 61)

Licenses (Page 71)

Starting installation (Page 73)

Modifying or updating installed products (Page 78)

Repairing installed products (Page 80)

Starting to uninstall (Page 82)

Installation log (Page 72)

3.7 Modifying or updating installed products

You have the option to modify installed products using the setup program or to update to a new version.

Requirement

- Hardware and software of the programming device or PC meet the system requirements.
- You have administrator privileges on your computer.
- All running programs are closed.

Procedure

To modify or update installed products, follow these steps:

1. Insert the installation medium in the relevant drive.

   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.

   The dialog for selecting the setup language opens.
3. Choose the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click the "Read Notes" or "Installation Notes" button.

   The help file containing the notes opens.
5. Once you have read the notes, close the help file and click the "Next" button.

   The dialog for selecting the installation variant opens.
6. Select the "Modify/Upgrade" option button and click the "Next" button.

   The dialog for selecting the product languages opens.
7. Select the check boxes of the product languages that you want to install. You can remove previously installed product languages by clearing the corresponding check boxes.

   Note

   Note that the product language "English" cannot be removed.
8. Click the "Next" button.

   The dialog for selecting the product configuration opens.
9. Select the check boxes of the components that you want to install. You can remove previously installed components by clearing the corresponding check boxes.

10. Click the "Next" button.

    Note

    Note that you cannot change the target directory because the existing installation is being modified.

    If changes to the security and permissions settings are required in order to install the TIA Portal, the security settings dialog opens.

11. To continue the installation, accept the changes to the security and permissions settings, and click the "Next" button.

    The next dialog displays an overview of the installation settings.

12. Click the "Modify" button.

    This starts the installation of the additional components.

    Note

    If the installation was successful, a message to this effect is displayed on the screen. If errors occurred during installation, an error message is displayed informing you of the type of errors.

    ---

13. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".

14. If the computer does not reboot, click "Exit".

Result

The existing installation has been modified on your computer.

See also

Notes on the system requirements (Page 61)

Licenses (Page 71)

Starting installation (Page 73)

Displaying the installed software (Page 77)

Repairing installed products (Page 80)

Starting to uninstall (Page 82)

Installation log (Page 72)

3.8 Repairing installed products

You have the option to repair installed products by completely reinstalling them using the setup program.

Requirement

- Hardware and software of the programming device or PC meet the system requirements.
- You have administrator privileges on your computer.
- All running programs are closed.

Procedure

To repair installed products, follow these steps:

1. Insert the installation medium in the relevant drive.

   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.

   The dialog for selecting the setup language opens.
3. Choose the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click the "Read Notes" or "Installation Notes" button.

   The help file containing the notes opens.
5. Once you have read the notes, close the help file and click the "Next" button.

   The dialog for selecting the installation variant opens.
6. Select the "Repair" option button, and click the "Next" button.

   The next dialog displays an overview of the installation settings.
7. Click the "Repair" button.

   This starts the repair of the existing installation.

   Note

   If the installation was successful, a message to this effect is displayed on the screen. If errors occurred during installation, an error message is displayed informing you of the type of errors.
8. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".
9. If the computer does not reboot, click "Exit".

Result

The installed products have been reinstalled.

See also

Notes on the system requirements (Page 61)

Licenses (Page 71)

Starting installation (Page 73)

Displaying the installed software (Page 77)

Modifying or updating installed products (Page 78)

Starting to uninstall (Page 82)

Installation log (Page 72)

3.9 Starting to uninstall

Introduction

Software packages are removed automatically by the setup program. Once started, the setup program guides you step-by-step through the entire removal procedure.

You have two options for removing:

- Removing selected components via the Control Panel
- Removing a product using the installation medium

Note

The Automation License Manager will not be removed automatically when you remove the software packages, because it is used for the administration of several license keys for products supplied by Siemens AG.

Removing selected components via the Control Panel

To remove selected software packages, follow these steps:

1. Open the Control Panel with "Start > Settings > Control Panel".
2. Double click on "Add or Remove Programs" in the control panel.

   The "Add or Remove Programs" dialog opens.
3. Select the software package to be removed in the dialog "Add or Remove Programs", and click "Remove".

   The dialog for selecting the setup language opens.
4. Select the language in which you want the setup program dialogs to be displayed and click the "Next" button.

   The dialog for selecting the products you want to remove opens.
5. Select the check boxes for the products that you want to remove and click the "Next" button.

   The next dialog displays an overview of the installation settings.
6. Check the list with the products to be removed. If you want to make any changes, click the "Back" button.
7. Click the "Uninstall" button.

   Removal begins.
8. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".
9. If the computer does not reboot, click "Exit".

Removing a product using the installation medium

To remove all software packages, follow these steps:

1. Insert the installation medium in the relevant drive.

   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.

   The dialog for selecting the setup language opens.
3. Choose the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click the "Read Notes" or "Installation Notes" button.

   The help file containing the notes opens.
5. Once you have read the notes, close the help file and click the "Next" button.

   The dialog for selecting the installation variant opens.
6. Select the "Uninstall" option button and click the "Next" button.

   The next dialog displays an overview of the installation settings.
7. Click the "Uninstall" button.

   Removal begins.
8. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".
9. If the computer does not reboot, click "Exit".

See also

Installation log (Page 72)

Notes on the system requirements (Page 61)

Licenses (Page 71)

Starting installation (Page 73)

Displaying the installed software (Page 77)

Modifying or updating installed products (Page 78)

Repairing installed products (Page 80)

3.10 Installing and uninstalling the migration tool

3.10.1 System requirements

System requirements for the migration tool

The following system requirements apply to the use of the migration tool:

- All products used to create the source project must be installed. The following products are supported:
  - SIMATIC STEP 7 V5.4 SP5 and STEP 7 V5.5
  - WinCC V7 SP1 or V7 SP2
  - WinCC flexible V1.3 SP2
- All optional packages needed to process the STEP 7 project are installed. For example, all HSPs for the devices used in the source project are required.

3.10.2 Installing the migration tool

Distribution of the migration tool

The migration tool is available for download from the Service & Support area of the Siemens website.

Normally, the migration tool is installed without the TIA Portal. Because the TIA Portal has its own integrated migration function, a separate installation of the migration tool is not necessary.

Procedure

To install the migration tool, proceed as follows:

1. Download the installation file from the Service & Support area on the Siemens website.
2. Run the downloaded file.

   The setup program for the migration tool will open.
3. First, select the language in which the setup should be displayed and click the "Next" button.

   The page for selecting the software language is displayed.
4. Since the migration tool is provided exclusively in English, you cannot choose any other language for the installation. Therefore, click "Next" to proceed to the next step.

   The page for selecting the product is displayed.

5. The migration tool consists solely of a software component. Therefore, the migration tool is already selected.

   To create a Desktop icon for starting the migration tool, select the check box "Create Desktop icon". Then click the "Next" button.

   The page for confirming the licensing terms is shown.

6. Click on an entry in the list of license terms to read the selected license term. If you agree with all license terms, select the check box "I accept the terms of the displayed license agreement". Then click the "Next" button.

   An overview of the installation is displayed.

7. Click the "Install" button.

   The installation is performed with the displayed settings.

3.10.3 Uninstalling the migration tool

The migration tool can be removed using the Control Panel.

Procedure

To remove the migration tool, follow these steps:

1. Open the Control Panel.
2. Double click on "Add or Remove Programs" in the Control Panel.

   The "Add or Remove Programs" dialog opens.

3. Select the "TIA Portal Migration Tool V11" entry in the "Add or Remove Programs" dialog, and click the "Remove" button.

   A confirmation prompt appears.

4. Click the "Uninstall" button to confirm this prompt.

   The migration tool will be removed.

Migrating projects 4

4.1 Migrating projects

Migration of existing projects

You can migrate projects from earlier automation solutions to the TIA Portal. Each time you migrate, a new project is created for the migrated data with which you can then work. Any projects already open are closed first.

The migration is then displayed in the table of the project history. From here, you have access to a log file that is created automatically for the migration.

Supported products for migration

The chapter "System overview STEP 7 and WinCC" includes information on the products that are available for the TIA Portal. In principle, all products listed there are supported by the TIA Portal during migration.

Any additional requirements that must be met depend on the initial products that were used and the currently installed products. For more information on the migration options for your products, you can, for example, refer to the Service & Support Internet pages and the documentation of your software products.

See also: Scaling of STEP 7 and WinCC (Page 23)

Procedure during migration

The migration process is divided into the following basic steps:

1. Preparing the initial project

   If the software that was used to create the initial project is not installed or is only partly installed on the programming device/PC together with the TIA Portal installation, you first need to convert the project to a special migration format with the file extension "AM11". To do this, install the migration tool on a PG/PC on which the required software is installed.

   Then, use the migration tool to convert the initial project, and copy the file to the programming device/PC on which the TIA Portal is installed.

   You can omit this step if the initial project and its associated software are on the same programming device/PC as the TIA Portal. In this case, you can directly migrate the source project.

2. Performing migration

   You perform the actual migration within the TIA Portal.

3. Checking the migration log

   A log is created for each migration, in which the result of the migration is stored. In this log, you can find information on project components that have been lost or modified, or, when applicable, the reasons why the migration of a project could not be performed. Check this log after migration. Immediately after completion of the migration, the log is displayed in the Inspector window. You can also open the log from the project history. If a migration is not possible, you can decide where the log will be saved.

4. Correcting the migrated project

   Because the configurations of the initial project may not always be completely compatible with the TIA Portal, not all configurations are transferred in identical form in the migrated project. You should therefore work through the points in the migration log systematically.

Excluding the hardware configuration from the migration

If you know at the outset that hardware used in the initial project is not supported by the TIA Portal, you can exclude the hardware configuration from the migration. In this case, only the software is migrated.

An unspecified device is then generated in the migrated project for the devices contained in the initial project. The hardware and network configurations and the connection are not migrated. You can convert the unspecified devices into suitable devices after the migration and create any network configurations and connections manually.

See also

Displaying the log file of the migration (Page 95)

Scaling of STEP 7 and WinCC (Page 23)

4.2 Preparing projects with the migration tool

4.2.1 Migrating projects with the migration tool

Preparation for migration

In many cases, a project you want to migrate is not located on the same programming device/PC on which the TIA Portal V11 is installed. Therefore, the initial project must first be converted to a compatible format for the migration. You then copy the migration file to the programming device or PC on which the current version of the TIA Portal is installed. There you can then read the file into the TIA Portal and create a project in the current file format of the TIA Portal.

Procedure for migration with the migration tool

To perform a migration without the initial software, the following preparatory steps are necessary:

1. Install the migration tool on the programming device/PC where the source project is located. To do this, download the installation file from the Service & Support area on the Siemens website.
2. Start the migration tool and use it to convert the source project to the migration file format with file extension ".ap11".

   For this step, make sure that all software needed to process the source project is installed on the programming device/PC. This also includes all necessary service packs, hardware support packages and all expansion software that is needed to process the initial project. If individual products are not installed it may not be possible to perform the migration or the migration may be incomplete.
3. Copy the migration file to the target system on which a current version of the TIA Portal is installed.

   Note that the target system must have been installed with all software needed to configure the complete set of devices contained in the migration.
4. Perform the migration within the TIA Portal, and specify the migration file as the source.
5. Following the migration, check the log file and work through the information provided there for the newly created project in a systematic manner. Read the information in the Inspector window with special care after the first compilation of the configuration.

Excluding the hardware configuration from the migration

If you know at the outset that hardware used in the initial project is not supported by the TIA Portal, you can exclude the hardware configuration from the migration. In this case, only the software is migrated.

An unspecified device is then generated in the migrated project for the devices contained in the initial project. The hardware and network configurations and the connection are not migrated. You can convert the unspecified devices into suitable devices after the migration and create any network configurations and connections manually.

See also

Migrating projects (Page 87)

Migrating projects (Page 92)

Calling the migration tool (Page 90)

Creating a migration file (Page 91)

4.2.2 Calling the migration tool

Starting the migration tool

A "Migration to TIA Portal V11" shortcut is always created during the installation in Start menu under "Siemens Automation > Migration Tool". Click this shortcut.

Alternatively, you can call the migration tool directly in Windows Explorer. The migration tool is saved to the following default folder during installation: "C:\Program Files\Siemens\Automation\Portal V11\Mig\bin". To start the migration tool, click the "Siemens.Automation.MigrationApplication.exe" file in this directory.

See also

Creating a migration file (Page 91)

4.2.3 Creating a migration file

The section below describes how you can use the migration tool to convert the initial project into a migration file that can be read by the TIA Portal. Following conversion, this file is transferred to the target system and migrated there.

You can specify whether the migration file should contain the entire project, including the complete hardware configuration and the associated software, or whether you want to migrate the software only.

Requirement

- The suitable, original software with a valid license is installed for all configurations used in the initial project.
- The initial project is not provided with access protection.
- The initial project must be consistent, otherwise problem-free migration cannot be assured.

Procedure

To create the migration file, follow these steps:

1. Choose the path of the source file for the migration in the "Storage Location (Path)" field.
2. Select the check box "Exclude hardware configuration" to migrate the software only.
3. Choose the path and the file name for the migration file in the "Intermediate file".
4. Click the "Migrate" button.

Result:

A migration file is created. Finally, copy this file to the target system and migrate this file in the TIA Portal.

See also

Migrating projects (Page 92)

Calling the migration tool (Page 90)

Migrating projects with the migration tool (Page 89)

4.3 Migrating projects

Requirement

- There is already a converted file in the format AM11 available or the original software with a valid license is installed for all the configurations in the initial project.
- The initial project is not provided with access protection.
- The initial project must be consistent, otherwise problem-free migration cannot be assured.

Read the additional information on the requirements in the help for the respective products installed.

Note

System hibernation during the migration

While a migration is running, the system should not be changed to the standby or hibernate mode. Otherwise the migration will be aborted.

Procedure

To migrate a project, follow these steps:

1. Select the "Migrate project" command in the "Project" menu.

   The "Migrate project" dialog opens.
2. Specify the path and the file name for the project to be migrated in the "Source path" field. Choose either a project in the AM11 migration format or in the format of the initial project.
3. Select the check box "Exclude hardware configuration" to migrate the software only.

   If you have selected a migration file that was created using the migration tool, the check box is disabled. In this case, you must specify if you wish to exclude the hardware configuration of the migration before the conversion with the migration tool .
4. Choose a name for the new project in the "Project name" box.
5. Choose a path in the "Target path" box where the new project will be created.
6. Enter your name or the name of another person responsible for the project in the "Author" field.
7. Enter a comment in the "Comment" box, if you require one.
8. Click "Migrate".

Result

The initial project is converted. The system outputs a message when the migration is complete. From here, you can directly open a log for the migration. The newly created project then opens in the project view. The migration log is also displayed in the Inspector window.

See also

Post-editing integrated projects (Page 152)

Migrating projects with the migration tool (Page 89)

Creating a migration file (Page 91)

4.4 Displaying the history of the migration

If a project was created by migration, the migration will be listed in the table of the project history. You can open the migration log in the table. The time of the migration is also shown.

Procedure

To display the migration in an overview table, follow these steps:

1. Select the open project in the project tree.
2. Select "Properties" in the shortcut menu of the project.

   The dialog with the properties of the project opens.
3. Select the "Project history" group in the area navigation.

   The overview table is displayed.

See also

Displaying properties of the project (Page 212)

4.5 Displaying the log file of the migration

A log file is created for each migration. The log file contains the following information:

- Migrated objects
- Modifications to objects made during migration
- Errors that occurred during migration

Procedure

To display the log file of the migration, follow these steps:

1. Select the open project in the project tree.
2. Select "Properties" in the shortcut menu of the project.

   The dialog with the properties of the project opens.
3. Select the "Project history" group in the area navigation.

   The overview table is displayed.
4. Click on the link to the log file in the "Log file" column.

   The log file is displayed in the Microsoft Internet Explorer.

See also

Migrating projects (Page 87)

4.6 Migrating STEP 7 projects

4.6.1 Migration of STEP 7 projects

Introduction

You can migrate a project from SIMATIC STEP 7 V5.4 SP5, or from SIMATIC STEP 7 V5.5 to be able to continue to use it in the TIA Portal. It is also possible to migrate integrated projects with devices from other software products.

Scope of migration

Generally, you can migrate all of the configurations from SIMATIC STEP 7 V5.4 SP5 or V5.5 which are also supported by the TIA Portal STEP 7 Professional. This includes, for example, the following devices and configurations:

- Devices of the S7-300 and S7-400 family.
- PROFIBUS configurations with connected distributed I/O. These include GSD-based slaves, I-slaves as well as direct data exchange.
- PROFINET configurations with distributed I/O. These include GSDML-based devices and I-devices.
- Network configurations
- Connections
- Blocks created in the programming languages LAD, FBD or STL, S7-SCL, S7-GRAPH
- PLC tags
- User-defined data types (UDT)
- Alarms and alarm classes
- Interrupts
- User-defined attributes (DUE) if they are supported by STEP 7 Professional V11
- User text libraries

Migrating software without hardware configuration

If individual devices are not supported by the TIA Portal, you can exclude the hardware configuration from the migration. In this case, only the software is migrated.

The devices are displayed in the new project as unspecified devices. Networks, connections and the entire hardware configuration are not imported. After migration, convert the unspecified devices to suitable devices again. Then network the modules again and restore all the connections manually.

See also

Reporting system errors (Page 105)

General notes on migrating program blocks (Page 105)

4.6.2 Requirements for the migration of STEP 7 projects

Various requirements for the software installed on the original PG/PC and for the initial project apply to the migration.

Requirements for the original PG/PC

The following requirements apply to the PG/PC:

- A licensed version of STEP 7 V5.4 SP5 or V5.5 must be installed.
- For all of the configurations used in the project, the corresponding additional software must be installed with a valid license, for example, optional packages.
- All HSPs for modules not included in the hardware catalog must be installed.
- All GSD/GSDML files used in the project must be installed.
- You must be logged on to the operating system with administrative rights.
- Either STEP 7 Professional V11 or the migration tool must be installed on the programming device/PC.

Requirements for the initial project

- The initial project must not be configured with access protection.
- The hardware must be consistent.
- The message number assignment must be set to CPU-oriented.
- It must contain no protected blocks with time stamp conflict.
- It must be possible to compile all programs and their sources without errors.
- All called blocks must be available in the block folder.
- The block folder must contain no blocks that are not called, in particular no instance data blocks.

Requirements for the programming device/PC with STEP 7 Professional V11

- A licensed version of STEP 7 Professional V11 or higher must be installed.
- For all of the configurations used in the project, the corresponding additional software must be installed with a valid license, for example, optional packages.
- All HSPs for modules that are not contained in the hardware catalog and all GSD/GSDML files used in the project must be installed.

4.6.3 Checking whether STEP 7 projects can be migrated

Before you start the migration, you can check if all necessary requirements for migration have been met.

Note

Note that the actual values are reset in the original project when the following steps are carried out. If this is not desired, first make a copy of the project and then carry out the check on the copy of the project.

Procedure

To check whether a project can be migrated, follow these steps:

1. Open the original project in SIMATIC STEP 7 V5.4 SP5 or V5.5. For a correct update, it is necessary that only the optional packages and modules, including HSPs, GSD and GSDML files, that are contained in STEP 7 Professional V11 are installed.
2. Open the individual stations. If no messages indicating missing components are displayed when the stations are opened, all the components necessary for migration are available.
3. Perform a block consistency check for each block container contained in the project.
4. Compile the entire project. If no errors are displayed during the compilation, the programs can be migrated.
5. Check whether all modules with an identical MLFB and the same firmware version are contained in the STEP 7 Professional V11 hardware catalog.
6. Check whether the CPU-oriented message number assignment is activated for each CPU.
7. Compile the project in NetPro. Here too, no errors should error.

4.6.4 Removing unsupported hardware components

If you have created configurations in the initial project for which not all of the right software is installed, then you can still migrate the project by manually deleting the unsupported configurations from the project. Alternatively, you can use the "Reorganize" function.

Procedure

To clean the original project, follow these steps:

1. Use an Installation of SIMATIC STEP 7 V5.4 or V5.5 that contains only the optional packages contained in STEP 7 Professional V11 and all required modules.
2. Save the project again and select the "With reorganization" option when saving. Any unsupported configurations will be removed from the project.

For modules that are only available in a new version or with a new firmware version in STEP 7 Professional V11, use the "Exchange object" function in the station to exchange an older module for one that can be migrated.

4.6.5 Special points to observe during migration

4.6.5.1 Device names

Station names

In SIMATIC STEP 7 V5.5, the name of an interface module is used as station name. In the TIA Portal, however, the name of the CPU will be used. For this reason, the station name is changed during the migration.

Names of modules and racks

In SIMATIC STEP 7 V5.5, the names of modules and racks do not have to be unique. In the TIA Portal, however, unique names are required. Therefore, the names of the modules and racks are incremented during the migration to make sure that unique names are assigned. This does not apply to PROFINET IO devices and PROFINET CPs. You must provide unique names for these yourself.

4.6.5.2 PROFINET IO configurations

Unique name assignment required

The names of PROFINET IO devices must be unique within the subnet. The same applies to the names of PROFINET CPs. For this reason, ensure that the name assignment is unique before the migration into the initial project or provide unique names after migration and before the first compilation.

Media redundancy with sync domains

If a sync domain with activated media redundancy is found in the initial project, the sync domain configuration is imported during migration. If, however, several sync domains with activated media redundancy are found in the initial project, they are removed and the project is migrated without the sync domains. In any case, you will receive a message which sync domains were imported.

In any case, use SIMATIC STEP 7 V5.5 to configure the media redundancy in the initial project.

Isochronous mode with PROFINET IO

Observe the following for PROFINET IO configurations in isochronous mode:

- IRT with high performance

  If you have configured IRT with high performance when planning the topology in the initial project, the option is changed to "RT" during migration. You will receive a message informing you of the change.

- Use of OB 61

  If the isochronous mode for PROFINET IO with OB 61 is activated in the original configuration in the STEP 7 project, it will be deactivated by the migration. The devices will still be imported. PROFINET IRT configurations with the option "high flexibility" without the use of OB 61 can be migrated.

CBA for IO controllers

If you have used the cyclical service CBA (Component Based Automation) in the initial project, it is deactivated during migration. You will receive a message informing you of the change.

Use of shared devices

If shared devices are configured in the initial project, they are removal during migration. You will receive a message informing you of their removal.

Transfer areas

If transfer areas for data exchange between the IO controller and I device are configured in the initial project, they are removed during migration. You will receive a message informing you when a transfer area is removed during migration.

Update groups for PROFINET IO

If necessary, also remove update groups from the STEP 7 project before you start the migration. If any update groups are still configured, then the migration can not be carried out.

I slaves and I devices

Only connected and consistent I slave and I device configurations can be migrated. If the configuration is inconsistent, then the configurations in question will be excluded from the migration.

4.6.5.3 PROFIBUS configurations

Isochronous PROFIBUS

It is possible to migrate isochronous PROFIBUS configurations. You should compile the project again after the migration, however, to rule out any possible inconsistencies. Also pay attention to slightly changed SDBs.

4.6.5.4 Connections

Connections from SIMATIC STEP 7 projects can usually be migrated if they are also supported by TIA Portal STEP 7 Professional V11.

Migratable types of connection

The following connection types can be migrated:

- S7 connection one-way
- S7 connection two-way
- Failsafe S7 connection
- TCP connection
- ISO-on-TCP connection
- ISO connection
- UDP connection
- E-mail connection
- FDL connection
- Point-to-point connection

FMS connections on the PROFIBUS are not accepted during the migration. Adapt the connections or modify the user program.

4.6.5.5 Multiprojects

Multiprojects in SIMATIC STEP 7

In SIMATIC STEP 7, you can organize projects into a multiproject if, for example, a project is very large or if several people are working on it. The subprojects may contain cross-project references, for example connections. All subprojects of a multiproject and the associated libraries are stored in the same directory.

Migration of parts of a multiproject

You can migrate the subprojects of a multiproject. To do this, select one of the subprojects during the migration. All of the devices and configurations contained in the respective subproject will be imported during the migration.

Observe the following when migrating a subproject:

- Cross-project connections

  Cross-project connections between subprojects are created as unspecified connections.
- Cross-project networking

  Information on networking of devices across project boundaries will be lost during the migration. Links between devices in the same subproject will be retained however.

Grouping a multiproject for the migration.

It is only possible to migrate an entire multiproject with all of its associated subprojects if you manually regroup the different subprojects prior to the migration. This is possible, for example, by copying the individual devices and pasting them back into a single project.

4.6.5.6 Messages

It is generally possible to migrate messages that you have created in SIMATIC STEP 7 projects.

Requirements

The prerequisite is that you have selected the CPU-wide assignment of message numbers in your SIMATIC STEP 7 project. If this is not the case, you must first switch the project to be migrated from the project-wide assignment of message numbers in SIMATIC STEP 7 to the CPU-wide assignment of message numbers.

Restrictions

Symbol-based messages cannot be migrated.

4.6.5.7 Reporting system errors

In principle, you can migrate system error messages that you have configured using SIMATIC STEP 7.

Restrictions

When migrating the "Report system errors" application, only the settings that were set by the user in the SIMATIC STEP 7 project will be migrated (e.g. basic settings, block numbers, block names).

The following elements are not migrated:

- SFM blocks
- SFM messages and the settings that were made to configure the message texts
- System text libraries

After migration, the settings can be manually adjusted once again. The missing elements will be recreated during the first compilation of the hardware configuration.

4.6.5.8 Migrating program blocks

General notes on migrating program blocks

You can generally migrate all blocks created in LAD, FBD, SCL, STL, and GRAPH. Please note the following:

Programs with absolute addressing

In the TIA Portal, symbolic operands are automatically declared for all absolute addresses. If absolute addressing was used in the STEP 7 program to be migrated, symbolic operands are also declared for the absolute addresses during the migration. This automatic declaration can lead to data type conflicts, particularly if the IEC check is activated in the TIA Portal. A data type conflict could occur, for example, if a 32-bit data value is declared as a DWORD type tag but a REAL data type tag is expected later in the program.

In this case, it is not enough to deactivate the IEC check. Instead, you must correct the declaration in the PLC tag table.

Symbolic names for blocks and user-defined data types (UDT)

In the TIA Portal, each block or UDT has a number and a name. It is therefore no longer necessary to declare symbolic names. If the migrated program contains symbol declarations for blocks or UDTs, they will be used as names after the migration.

Blocks from STEP 7 libraries

In the TIA Portal, standard functions and function blocks are not provided in libraries. Instead, they are available as instructions in the "Instructions" task card. The instructions are sorted by function and can be found under their symbolic name.

If calls of standard functions or function blocks are contained in the migrated program, they are replaced during the migration by the instructions that correspond to the original standard function or the original standard function block.

If the library block is no longer supported in the TIA Portal, you have the following options:

- Replace the library block with a compatible instruction. In this case, you will receive a message informing you that you have to compile the program after migration.
- If there is no compatible instruction, the block is migrated as user block. It is then available as know-how protected block in the folder "Program blocks" with the extension "_LF" (legacy function).

User-defined block libraries

User-defined block libraries are no longer available in the familiar form in the TIA Portal. However, you can migrate user-defined libraries by integrating these into a project prior to migration and then migrating the project. After this, you can copy these from this project in the TIA Portal and use them.

Instructions in a modified display

The display of some of the LAD, FBD, STL, SCL and GRAPH commands in the TIA Portal differs from the display in earlier versions of STEP 7. Thus, for example, the mathematical functions and the comparators are no longer specific for data types. Instead, there is a central instruction, which can be used for all types of data. The "ADD_I" command is no longer permitted, for example; the instruction "ADD" is used instead.

Some other commands also have a new display in the TIA Portal, e.g. edge commands, word logic operations, conversions, IEC timers, IEC counters, etc.

The commands will be converted for the new display during the migration.

Stricter IEC check

Stricter rules for checking data type compatibility will be used in the TIA Portal. In addition, errors that can lead to runtime errors are already detected during the syntax check. It is no longer permitted to write your own input parameters or to read your own output parameters in functions (FC). Observe the following rules to avoid syntax errors:

- Only use tags with compatible data types when transferring the parameters.
- In comparison instructions or arithmetic instructions, only use tags with compatible data types.
- You may not write to input parameters or read from output parameters.

For more information on the block interface and on data type conversion with IEC check, refer to "See also".

Peripheral access ID ":P"

In the TIA Portal, the access ID ":P" is used for direct addressing of the peripherals. The following notation is not permitted, for example:

```
%PEW3
```

The following notation is used instead:

```
%EW3:P //absolute display
MyTag:P //symbolic display
```

The accesses are converted into the new notation during the migration. However, symbolic names, which were declared to be peripheral tags in the original program, cannot be accepted. New declarations will be created instead. For additional information on peripheral access, refer to "See also."

No distinction is made between upper and lower case for jump marks

No distinction is made between upper and lower case for jump marks in the TIA Portal. If the source program contains jump labels that differ only through upper/lower case, these will be converted into unique jump labels. The log file provides information on the modified jump labels.

Migration of global data blocks

The display and handling of data values in data blocks is different in the TIA Portal than in SIMATIC STEP 7. In global data blocks which are not derived from a high-level object, for example a UDT, the tags are always assigned the default value of the data type as the default value, for example FALSE for the data type BOOL. This default value is not editable. If offline initial values were assigned in the declaration view, they are not imported during the migration. If you require user-specific default values in the program, use a global data block based on a PLC data type. You can assign user-specific default values for tag values in a PLC data type.

For additional information on data values, refer to "See also".

Migration of LAD/FBD/STL programs

LAD, FBD and STL blocks can run after the migration without being recompiled or reloaded.

Know-how protection

Blocks that were know-how protected before the migration remain know-how protected after the migration. Because the sources are not migrated, the know-how protection cannot be canceled. The blocks can thus not be opened or edited. It is possible, however, to load the blocks into the CPU and to run them.

The following options are available for disabling the a block's know-how protection:

- Prior to the migration, remove the KNOW_HOW_PROTECT attribute from the source and create a block without know-how protection from this. Then migrate this block.
- Import the source into the TIA Portal and create a block without know-how protection from this. Then use the "Edit > Know-how protection" menu command to provide the block with a password.

Switch-over from LAD/FBD to STL

As determined by the system, programs that were created in LAD or FBD and contain jump operations to subsequent networks will be displayed in STL after the migration. In the block properties, however, the language is still set as LAD or FBD.

In the block properties, change the program again to STL. Then reset the language to LAD or FBD.

Migrating GRAPH programs

Requirement

The prerequisite for migrating GRAPH blocks is to have the "S7-GRAPH" optional package, version 5.3 SP6 or later, installed on the original device.

GRAPH-specific block settings

The GRAPH-specific block settings have been significantly reduced in the TIA Portal. The following table gives an overview of how the original block settings will be transferred to the TIA Portal.

Setting in the initial project		Setting in the TIA Portal
FB parameter	Minimum	Existing parameters will be migrated
	Standard	
	Maximum	
	Definable	
Interface description	Minimized memory space	Generate DB with minimized memory space
	Structure fields	Individual structures are always created
	Individual structures	
Run capability	FC70/71	FC 73 if the option "Generate DB with minimized memory space" is active; otherwise FC 72
	FC72	
	FC73	
Criteria analysis in the DB		Setting not relevant in the TIA Portal
Skip steps		Skip steps
Acknowledgment of errors required		Acknowledge supervision errors
Synchronization		Always activated in the TIA Portal
Permanent processing of all interlocks in manual mode		Permanent processing of all interlocks in manual mode
Disable operating mode selection		Disable operating mode selection
Safeguarded switching behavior		Always activated in the TIA Portal
Warnings	None	Always activated in the TIA Portal
	Any	

Setting in the initial project	Setting in the TIA Portal
Activate messages	Activate messages
Messages with WR_USMSG	Messages in the TIA Portal always with ALARM_SQ/ALARM_S

Step name extension

Step name extensions are not supported in the TIA Portal. They are converted to step-specific text during the migration.

Compiling the migrated block

GRAPH blocks must be recompiled and reloaded after the migration.

The following errors may occur when compiling:

- Data type conflicts may occur due to the stricter IEC check. In this case, change the declaration.
- Since the "structure fields" setting for the interface is no longer available in the TIA Portal, interface elements can no longer be addressed with G7S[1].X, for example. Access to the element must therefore be corrected, e.g. to STEP1.X.

Changing the interface of the GRAPH block

The changed settings can lead to interface changes of the block. Therefore, it may be necessary to regenerate the instance DB or to update block calls.

Migrating SCL programs

The prerequisite for migrating SCL blocks is to have the "S7-SCL" option package, version 5.3 SP5 or later, installed on the original device.

SCL blocks must be recompiled and reloaded after the migration.

Basic procedure for the migration

A complete migration of SCL blocks is only carried out if the associated sources exist in the initial project.

The table below gives you an overview of the basic procedure.

Existing in the initial project	Existing after migration
SCL blocks with source	Editable SCL blocks
Know-how protected SCL blocks with source	Editable SCL blocks. After the migration, the blocks no longer have know-how protection but can be re-protected, if necessary.
SCL blocks without source	Know-how protected SCL blocks
Know-how protected SCL blocks without source	Know-how protected SCL blocks

SCL-specific compiler options

Only the compiler options that are defined directly in an SCL source are adopted as block properties in the TIA Portal during migration. If no compiler options are defined in the original SCL source, the options in the properties of the migrated block are deactivated.

Compiler options defined as settings in the SCL Editor or in an SCL compile control file are not migrated.

The following table gives an overview of the compiler options that are transferred as block property to the TIA Portal.

Compiler option in the SCL source	Block property in the TIA Portal
scl_monitorarraylimits	Check ARRAY limits
scl_createdebuginfo	Create extended status information
scl_setokflag	Set ENO automatically

Symbolic constants as ARRAY limits

In the TIA Portal, it is not possible to use symbolic constants as ARRAY limits. For example, the following notation would not be permitted:

```
ARRAY [min..max]
```

Only direct value inputs are possible as ARRAY limits, e.g.:

```
ARRAY [1..5]
```

During the migration, symbolic ARRAY limits are converted into direct values. For additional information on symbolic constants, refer to "See also".

New EN/ENO mechanism

SCL uses the EN/ENO mechanism of the TIA Portal. For this reason, all usages of the OK flag are replaced by ENO during the migration. The points where ENO was used in the original program are highlighted. You must check these points after the migration and adapt them to the new mechanism.

For additional information on the EN/ENO mechanism of the TIA Portal, refer to "See also".

Operator "DIV"

The "DIV" operator is no longer available in the TIA Portal. All usages of "DIV" will be converted to the standard notation "/" during the migration.

Nested ARRAYs

In the TIA Portal it is not possible to nest ARRAYs. The following declaration is not permitted, for example:

```
ARRAY[1..5] OF ARRAY[0..3] OF INT
```

Nested ARRAYs are converted into multi-dimensional ARRAYs during migration. The example would look as follows after the migration:

```
ARRAY[1..5, 0..3] OF INT
```

Declaration of jump labels (LABEL)

The declaration of jump labels is not possible in the TIA Portal. The following declaration would not be accepted from the initial project, for example:

```
LABEL
LABEL1, LABEL2, LABEL3 ;
END_LABEL
```

Set jump labels, however, will be retained in the program code and can be used for GOTO operations.

Declaration of symbolic constants in the block interface

In the TIA Portal, symbolic constants cannot be declared in the block interface. If symbolic constants are declared in the original program, they will be converted into global constant declarations and will be available in the PLC tag table. Name conflicts are handled as follows during conversion:

- If both constants have the same value, a global constant will be created with this value.
- If the values differ, then the second constant will be renamed. The renaming will be consistently carried out at all points of use.

Indexed peripheral accesses

In the TIA Portal, the syntax with rounded parentheses is used for indexed peripheral accesses. The following notation is not permitted, for example:

```
PEB[1]
```

The following notation is used instead:

```
EB(1):P
```

The accesses are converted into the new notation during the migration. For additional information on peripheral access, refer to "See also."

Indexed memory accesses

In the TIA Portal, the syntax with rounded parentheses is used for indexed memory accesses. The following notation is not permitted, for example:

```
EB[2]
```

The following notation is used instead:

```
EB(2)
```

The accesses are converted into the new notation during the migration. For additional information on addressing, refer to "See also."

Indexed access to data blocks

In the TIA Portal, the syntax with rounded parentheses is used for indexed data block accesses. The following notation is not permitted, for example:

```
DB100.DW[5]
```

The following notation is used instead:

```
DB100.DW(5)
```

The accesses are converted into the new notation during the migration. For additional information on addressing, refer to "See also."

Absolute access to data blocks

For an absolute access, the absolute designator of the data block must be used. Access using the symbolic designator is not permitted in the TIA Portal.

The following notation is not permitted, for example:

```
DB100.DW3
```

The following notation is used instead:

```
%DB100.DW3
```

During the migration, the marker "%" will be added for a detected absolute data block access. For additional information on addressing, refer to "See also."

Logarithmic function "EXPD"

The logarithmic function "EXPD" is no longer available in the TIA Portal. All usages of "EXPD" are converted to the standard notation "10**" during the migration.

Floating point numbers - exponential display

The exponential notation of floating point numbers without a decimal point is no longer permitted in the TIA Portal. The following notation is not permitted, for example:

```
3E10
```

The following notation is used instead:

```
3.0E10
```

During the migration, all usages of "3E10" will be converted to the standard notation "3.0E10". For additional information on floating point numbers, refer to "See also".

Blocks from STEP 7 libraries

The functions FC1 to FC40 from the STEP 7 classic library "IEC standard functions" are no longer available in the TIA Portal. Calls to the standard functions are converted to calls to the corresponding extended instructions during the migration. If there is no clear conversion possibility, a system error is issued.

In this case, change the SCL block after the migration and use SCL instructions or operators instead of IEC standard functions.

Example:

Use S5TIME_TO_TIME instead of S5TI_TIM (FC 33)

```
"TAG_TIME" := S5TIME_TO_TIME ("TAG_S5TIME");
```

Use the relational operator <> instead of NE_DT (FC 9):

```
IF #D1 <> #D2 THEN "MyOutput" : = 10;

END_IF;
```

4.7 Migrating WinCC flexible projects (Basic)

4.7.1 Basics (WinCC flexible)

4.7.1.1 Migration (WinCC flexible)

Introduction

You can continue to use projects in WinCC from WinCC flexible. The following version of WinCC flexible is supported:

- WinCC flexible 2008 SP2

The following sections describe the operating devices that are supported and the required conditions for a successful migration.

Projects from ProTool and from earlier versions of WinCC flexible cannot be migrated directly to WinCC. If you wish to continue to use such projects in WinCC, you must first migrate them to a supported version of WinCC flexible.

See also

Object support during migration (WinCC flexible) (Page 122)

Projects from Migrating WinCC flexible projects (WinCC flexible) (Page 117)

Compiling and loading a migrated project (WinCC flexible) (Page 119)

Migrating runtime data (WinCC flexible) (Page 132)

Migrating integrated projects (WinCC flexible) (Page 135)

Supported HMI devices (WinCC flexible) (Page 120)

Migration of data types (WinCC flexible) (Page 139)

4.7.1.2 Basics on migration (WinCC flexible)

Introduction

During migration, the project data from a WinCC flexible project are converted to the new format of WinCC. The data will not be evaluated to see if they are consistent in the project you want to migrate. If errors or warnings are output in a source project during compilation, these will not be resolved as part of the migration. This means you should be able to compile the project without errors prior to migration. Note the scope of a project during migration. The features of WinCC apply for migration. See Engineering system (Page 2848) for additional information.

Unique object names

The objects are clearly identified by the folders in which they are contained in WinCC flexible. Screen elements in groups are clearly identified by the group name.

In WinCC, an object name must be unique within an HMI device. The name of screen elements must be unique within a screen.

The uniqueness of the name is verified during migration. If a name is not unique according to the new rule, the object in question will be renamed. A renamed object will receive the suffix "#Mign", where "n" stands for a sequential number.

Example:

In WinCC flexible, tags located in different folders may have the same name. In WinCC, the tag name must be unique on the configured HMI device. This means tags with the same name from different folders will be renamed during migration.

Tags are renamed as follows:

Before migration	After migration
Folder_1/Tag_1	Folder_1/Tag_1
Folder_1/Tag_2	Folder_1/Tag_2
Folder_2/Tag_1	Folder_2/Tag_1#Mig1
Folder_2/Tag_2	Folder_2/Tag_2#Mig1
Folder_3/Tag_1	Folder_3/Tag_1#Mig2
Folder_3/Tag_2	Folder_3/Tag_2#Mig2

Affected objects

The following objects are renamed if necessary:

- Screens
- Screen objects
- Recipes
- Tags

Cancelling migration

The migration is cancelled in the following cases:

- If the project to be migrated is opened in the engineering system or in Runtime.
- If not enough memory space is available on the hard disk to create a a copy for migration of the project.
- If the migration cannot address the project database due to problems with the installed SQL-Server.
- If the migration cannot address the project database due to missing user authorization.
- If you select the "*.hmi" file for the migration in an integrated project. You must select the "*.s7" file for the migration in an integrated project.
- If the project was created with a version not supported by the migration.

Saving the project in the migration format

You do not have to execute the migration of a WinCC flexible project completely on the PC on which the project is available. You can prepare the migration by saving the project in the migration format. The migration tool is available for saving a WinCC flexible project in the migration format. The migration tool exports the engineering data from the WinCC flexible project and saves the data in the migration format "*.AM11".

For the actual migration, copy the data in the migration format to a PC on which the TIA-Portal is installed.

See the section "Auto-Hotspot" for more information on the migration tool.

See also

Engineering system (Page 2848)

4.7.1.3 Projects from Migrating WinCC flexible projects (WinCC flexible)

Introduction

When you migrate a project, data from a WinCC flexible project is loaded into a new project for WinCC. A new project is therefore created automatically for project migration. You cannot migrate to an existing project.

The migration can be started in both the Portal view and the Project view.

You should only migrate a project in a newly started TIA portal.

Information on the migration of integrated project can be found in the Chapter Migrating integrated projects (WinCC flexible) (Page 135).

If you only want to save the project in migration format, you can use the migration tool. See Basics on migration (WinCC flexible) for additional information.

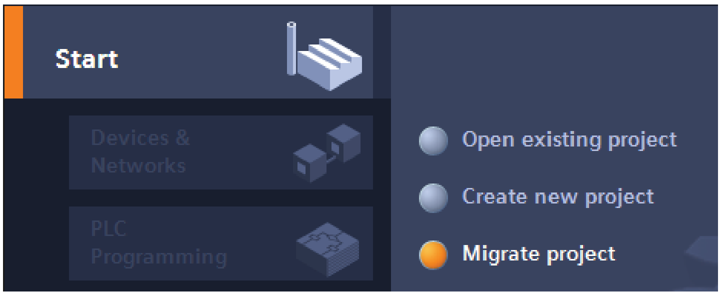
Requirement

- A project from WinCC flexible is available.
- The project is not open in WinCC flexible.
- The TIA portal is newly started.

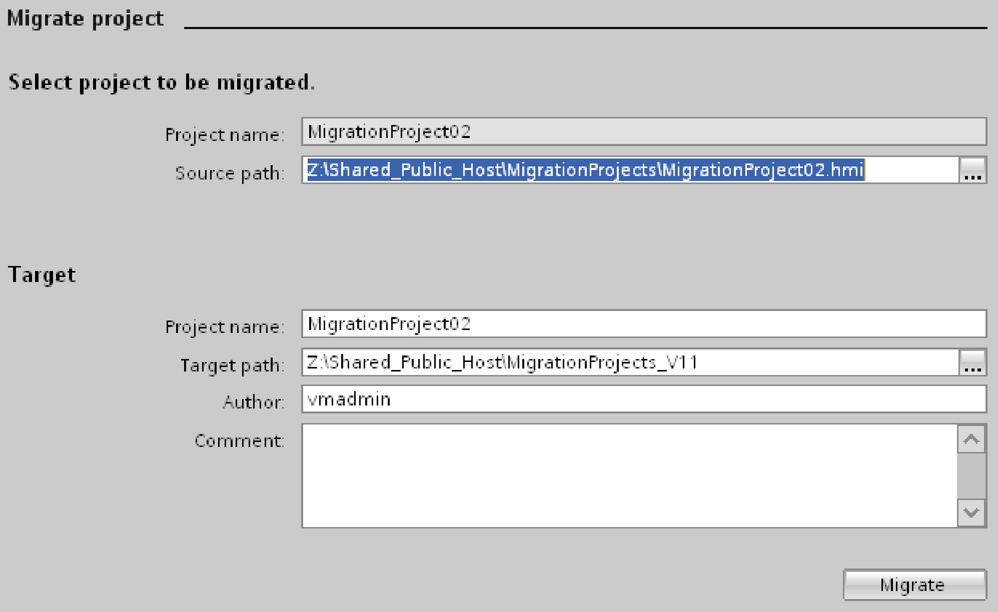
Procedure

Migrate a project in the Portal view as follows:

1. Select the action "Start > Migrate Project".



2. In the "Source path" box, navigate to the project you want to migrate.



Migrate project

Select project to be migrated.

Project name: MigrationProject02

Source path: Z:\Shared_Public_Host\MigrationProjects\MigrationProject02.hmi

Target

Project name: MigrationProject02

Target path: Z:\Shared_Public_Host\MigrationProjects_V11

Author: vmadmin

Comment:

3. Select the WinCC flexible project file "*.hmi".
4. Change the information for the project to be created, if necessary. For example, change the project name or project path. The data to be migrated is created in the new project.
5. Click "Migrate".
   A new project is created and migration of the data is started:
   - The Project view opens.
   - The progress of the migration is shown in a migration window.
   - Warnings and errors about the migration process are displayed in the Inspector window under "Info > General".
   - All information about the migration is saved in a log file.
   - A message is displayed upon completion of the migration. The message contains a link that you can use to open the log.
6. Once migration is completed, save the project.

When migration is complete, you will find a newly created device for each migrated HMI device in the project tree. These devices contain the migrated data, such as screens, alarms and tags.

Opening the migration log at a later point in time

The migration log is saved together with the migrated project. You can view the log at a later point in time. Open the log as follows:

1. Select the project in the project tree.
2. Select the "Properties" command in the shortcut menu.
3. Click "Project History" in the "Properties" dialog.
4. Click on the log file.
   The migration log opens.

See also

Migrating integrated projects (WinCC flexible) (Page 135)

4.7.1.4 Compiling and loading a migrated project (WinCC flexible)

Compiling a migrated project

Once you have successfully migrated a WinCC flexible project, you need to recompile it before loading it to the HMI device. The project will only compile successfully if it was capable of error-free compiling prior to migration.

If errors occur during compilation of the migrated project, they have to be eliminated.

Once compiling is successfully completed, load the project to the HMI device.

Settings for download to the HMI device

The settings for loading the HMI device are not included in the migration. Once you have migrated the project, you must configure the settings for loading.

Select the HMI device in the project tree and select "Loading in device > Software (complete loading)" from the shortcut menu.The dialog "Advanced Loading" is opened. Configure the required settings for the interface. Click the "Load" button. The project is recompiled and the dialog "Load preview" is opened.

Expand the "Overwrite" entry and verify the settings for the following options:

- Would you like to overwrite the existing user administration data from this device
- Would you like to overwrite the existing recipe data on HMI system

Configure the options as you want to use them in the project in the future. Subsequently, load the project to the HMI device.

4.7.2 Migrating engineering data (WinCC flexible)

4.7.2.1 Supported HMI devices (WinCC flexible)

Introduction

Note that WinCC only supports the following HMI device types when migrating projects from WinCC flexible:

- KTP400 Basic mono PN
- KTP400 Basic mono PN Portrait
- KTP600 Basic DP
- KTP600 Basic DP Portrait
- KTP600 Basic PN
- KTP600 Basic PN Portrait
- KTP600 Basic mono PN
- KTP600 Basic mono PN Portrait
- KTP1000 Basic DP
- KTP1000 Basic PN
- TP1500 Basic PN

WinCC only supports the functions provided by these HMI device types.

If your WinCC flexible project contains an HMI device that is not supported by WinCC, then the migration process will abort. To migrate the project, you must change the HMI device in WinCC flexible to a HMI device type supported by WinCC.

There may be some functions in a WinCC flexible project that are not supported by a Basic Panel, for example, because the device type has been switched. These unsupported functions are not migrated.

Adaptations before migration

If the HMI device has changed in the project being migrated, the project needs to be recompiled before migration. The compilation process will adjust the size of the screens and screen elements.

See also

Object support during migration (WinCC flexible) (Page 122)

Migration (WinCC flexible) (Page 114)

Migration of alarm classes and alarm groups (WinCC flexible) (Page 126)

Migration of language-dependent contents (WinCC flexible) (Page 129)

Migrating libraries (WinCC flexible) (Page 131)

Migration tags (WinCC flexible) (Page 125)

Changes of values of object properties by the migration (WinCC flexible) (Page 124)

4.7.2.2 Object support during migration (WinCC flexible)

Introduction

When migrating projects from WinCC flexible, all configuration data involving an HMI device supported by WinCC will be migrated. Basically, all object types and functions that are available and can be mapped to the new project environment will be fully migrated.

Some global object types are not migrated, for example, dictionaries and global libraries.

Supported object types

The following object types are supported for migration:

- Animations
- Scheduler
- User administration
- Area pointer
- Screens
- Screen template
- Data types
- Function lists
- Graphics lists
- Display and operating elements
  Migration supports all display and operating elements available on the supported HMI devices.
- Alarms
- Alarm classes
- Alarm groups
- Project library
- Project languages
- Recipes
- Runtime languages
- Runtime scripting
- System functions
- Texts
- Text lists
- Tags
- Connections

Unsupported object types

The following object types are not supported by migration:

- Global libraries
- Dictionaries
- Project versions
- Change log

Migration of the screen template

WinCC offers an extended concept for working with screen templates. WinCC offers a global screen and several templates for each device. During migration of a template from WinCC flexible, the objects contained there and the properties configured in the template are migrated to different templates of WinCC.

The following objects are migrated to the "global screen" of WinCC:

- Alarm window
- Alarm indicator
- Function keys of HMI devices with function keys

All other objects and properties are migrated to a template of WinCC.

The connection of the objects and properties to the respective template is automatically adapted.

Migration of system functions

The names of some system functions have changed in WinCC.

System functions which have changed their names are renamed.

This concerns the following system functions:

Function name in WinCC flexible	Function name in WinCC
IncreaseValue	IncreaseTag
DecreaseValue	DecreaseTag
SetValue	SetTag

See also

Supported HMI devices (WinCC flexible) (Page 120)

Changes of values of object properties by the migration (WinCC flexible) (Page 124)

4.7.2.3 Changes of values of object properties by the migration (WinCC flexible)

Introduction

The standardization of object properties from WinCC V7 and WinCC flexible requires changes to the object properties during the migration process. The migration calculates the changes in such a way that the representation of the objects after migration is the same as prior to migration. Changes made during migration result in different units of measurements and values in the configuration for some object properties.

Migrating the font settings of an object

In WinCC V7 and WinCC flexible, the unit of measurement "point" is used to denote the size of the fonts used for an object. In WinCC, the unit of measurement "pixel" is used to denote the size of the fonts used for an object. During migration, the font size is converted accordingly to ensure that the representation of the font is the same size at zoom level 100%. The different units of measurement result in changes to the numerical values for the font sizes after migration.

Example:

Font style before migration	Font style after migration
Arial 10 points	Arial 13 pixels
Arial 16 points	Arial 21 pixels
Tahoma 10 points	Tahoma 13 pixels
Tahoma 16 points	Tahoma 21 pixels

Migration of object margins

In WinCC flexible, some objects permit the entry of values <0 and >127 for setup of the object margins for the configuration of the representation. In WinCC, the range of values for object margins is limited to values between 0 and 127. The migration changes values <0 to the value "0" and values >127 to the value "127".

See also

Supported HMI devices (WinCC flexible) (Page 120)

Object support during migration (WinCC flexible) (Page 122)

4.7.2.4 Migration tags (WinCC flexible)

Introduction

You need to make some special considerations when migrating tags. The following aspects should be distinguished:

- Migrating data types of tags
- Migrating internal tags
- Migrating external tags
- Tag names

Migrating data types

WinCC features some other data types and uses different data type names than WinCC flexible. When migrating a relevant tag, the data type from WinCC flexible is mapped to the corresponding data type in WinCC. You can find additional details on this in the section "Migration of data types (WinCC flexible) (Page 139)".

Migrating tags

Tags are always fully migrated. Only the data type names and tag names may change due to migration.

Migrating names of tags

In WinCC flexible, tags located in different folders can have the same name. In WinCC, the tag name must be unique on the configured HMI device. This means tags with the same name from different folders will be renamed during migration. You can find additional details on this in the section "Basics on migration (WinCC flexible) (Page 115)".

See also

Basics on migration (WinCC flexible) (Page 115)

Migration of data types (WinCC flexible) (Page 139)

Supported HMI devices (WinCC flexible) (Page 120)

4.7.2.5 Migration of alarm classes and alarm groups (WinCC flexible)

Changing the names of alarm classes

In contrast to WinCC flexible, the names of the predefined alarm classes are not dependent on the user interface language currently in use. During migration, the names of the alarm classes are assigned as follows:

WinCC flexible	WinCC
Error	Alarms
System	System
Warnings	Events

The names of the alarm classes can be changed as necessary after migration.

Migrating alarm groups

Migration will migrate only those alarm groups actually in use.

Alarm groups with an ID from 1-31 will be migrated 1:1.

A corresponding alarm group is created in WinCC for each alarm class in the system. These alarm groups created by the system are assigned IDs beginning with the number 32 and consecutively incremented. The 4 pre-defined message classes in every WinCC project are automatically given IDs 32-35 by their alarm groups. Additionally created alarm group and an additional ID is assigned to each user-defined alarm class. Therefore, the IDs for alarms groups with IDs > 31 may be changed after migration. This step also changes the assignment of the alarm group names to the IDs.

Example:

In the example, you can see the assignment of the IDs in WinCC for the migration.

Alarm groups	ID in WinCC flexible	ID in WinCC	
Alarm group 1-16	1-16	1-16	Default for alarm groups from system alarms
Alarm group 17-31	17-31	17-31	Custom alarm groups
		32-35	Default in WinCC for alarm groups of predefined alarm classes.
Alarm group 32	32	36	Changed assignment of ID to alarm group in WinCC
Alarm group 33	33	37	Changed assignment of ID to alarm group in WinCC

Also note:

When migrating alarm groups that supposedly have the same group name, the migration adapts the name. This occurs, for example, when a group name contains a space at the end of the name. The migration deletes all existing spaces at the end of names. If two groups obtained the same group names due to this deletion, the migration adds the suffix "# Mign" to the group name of the following alarm groups, where "n" stands for a sequential number.

Example:

The following alarm groups exist in WinCC flexible:

"AlarmGroup_18"

"AlarmGroup_18 " - group name contains one space

"AlarmGroup_18 " - group name contains two spaces

"AlarmGroup_18" is the alarm group with the highest number.

Result after migration:

"AlarmGroup_18"

"AlarmGroup_18#Mig1"

"AlarmGroup_18#Mig1.1"

Changing the names of alarm classes

In contrast to WinCC flexible, the names of the predefined alarm classes are not dependent on the user interface language currently in use. During migration, the names of the alarm classes are assigned as follows:

WinCC flexible	WinCC
Error	Errors
System	System
Warnings	Warnings

The names of the alarm classes can be changed as necessary after migration.

Display of ALARM_S messages and SIMATIC SFM messages

In WinCC flexible you can activate the display classes for ALARM_S messages in integrated projects. In WinCC flexible, you activate the display of SIMATIC SFM messages via a separate setting. The separate setting for activating the display of SIMATIC SFM messages is not required in WinCC. You control the display of SIMATIC SFM messages, and also the display of ALARM_S messages in WinCC only by activating the corresponding display class.

The changed concept may cause the display of messages to change following migration.

If all the display classes for ALARM_S messages are activated and the display of SIMATIC SFM messages is deactivated in the WinCC flexible project , ALARM_S messages and SIMATIC SFM messages are displayed following migration.

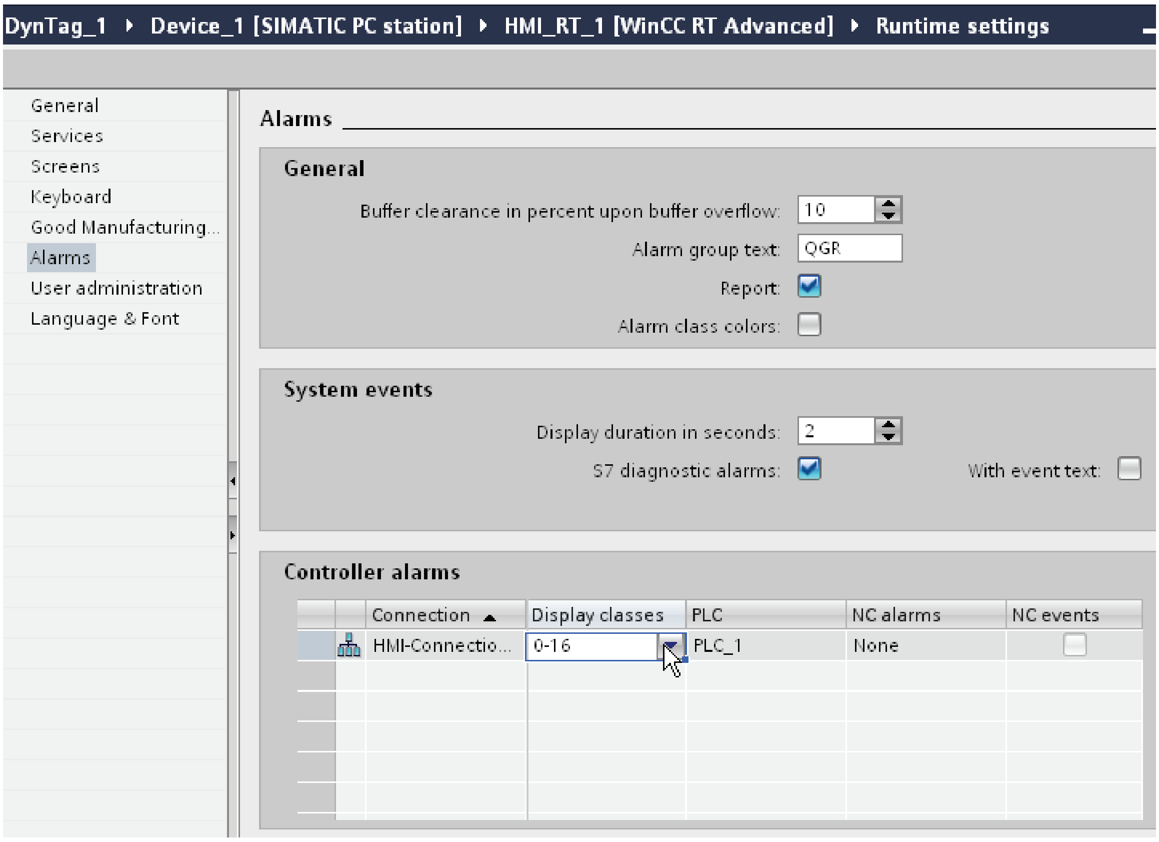
To ensure that only ALARM_S messages are displayed following migration, you have to assign the SIMATIC SFM messages to an unused display class after migration to STEP 7. You then have to deactivate this display class in WinCC.

If all the display classes for ALARM_S messages are deactivated and the display of SIMATIC SFM messages is activated in the WinCC flexible project , ALARM_S messages and SIMATIC SFM messages are not displayed following migration.

To ensure that only SIMATIC SFM messages are displayed following migration, you have to assign the SIMATIC SFM messages to an unused display class after migration to STEP 7. You then have to activate this display class in WinCC.

The display class is dependent on the settings in STEP 7. The default setting for SIMATIC SFM messages in Step 7 is the display class "0". To activate the display in WinCC, the display class "0" must be activated.

You activate the display class in WinCC in the Runtime settings of the respective HMI device in the "Messages" category.



Alarms

General

Buffer clearance in percent upon buffer overflow: 10

Alarm group text: QGR

Report: ☑

Alarm class colors: □

System events

Display duration in seconds: 2

S7 diagnostic alarms: ☑ With event text: □

Controller alarms

Connection	Display classes	PLC	NC alarms	NC events
HMI-Connectio...	0-16	PLC_1	None	□

See also

Supported HMI devices (WinCC flexible) (Page 120)

4.7.2.6 Migration of language-dependent contents (WinCC flexible)

Introduction

WinCC offers the same options for configuring projects in different languages as those available in WinCC flexible. All languages supported by WinCC are included in the migration of a project.

Migrating language-dependent content

The following language-dependent content is migrated:

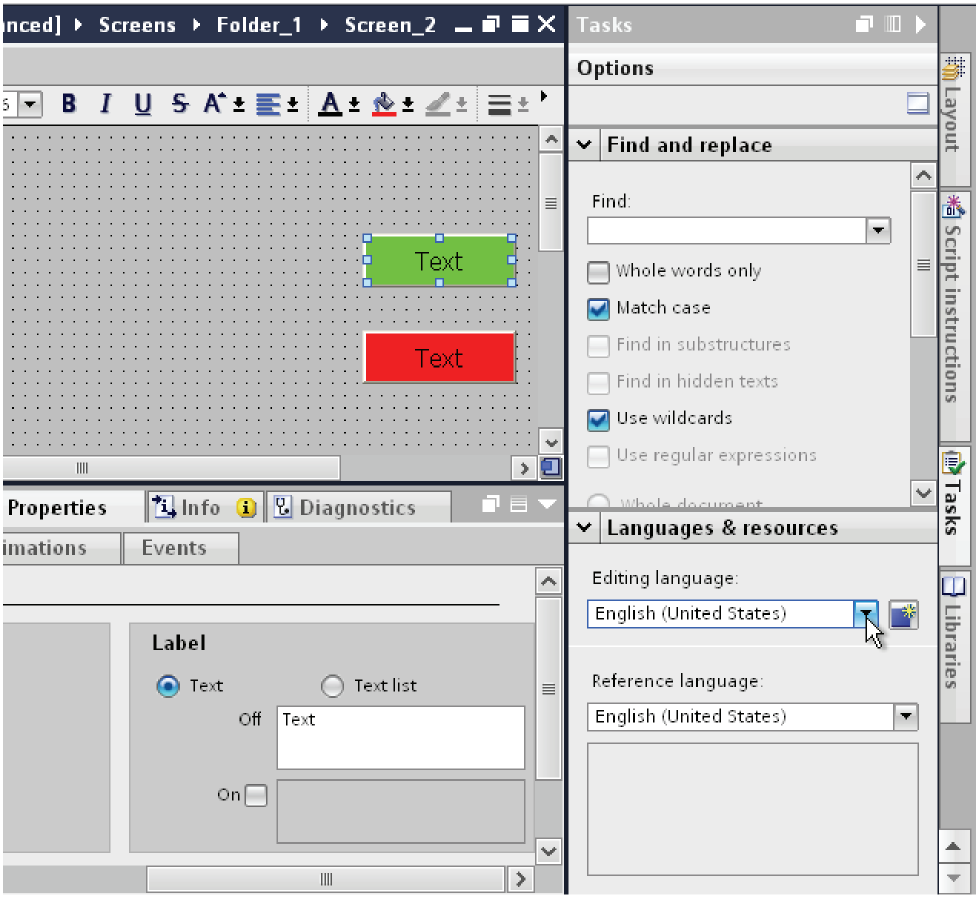
- Project languages
- Project texts
- Fonts for display in runtime
- Language-dependent graphics

You need to consider the following when migrating language-dependent content:

- The operating system on the PC performing the migration must support the project languages used in the project.
- The fonts used for runtime display must be installed on the PC performing the migration.
- Dictionaries are not supported by the migration.

Editiing language of integrated projects following migration

During migration of an integrated project, the project components to be migrated from STEP 7 and WinCC flexible also bring their respective settings for the editing language. In WinCC there is only one editing language for all project components. Migration activates for the mgrated project the editing language which was set in STEP 7 prior to migration. If this setting is not the same as the setting from WinCC flexible, the configured texts are no longer visible in WinCC. No text is displayed at the usage locations, or only the entry "Text" can be seen. To make the texts visible, you must change the editing language. Click the "Tasks" taskcard at the right-hand edge of the TIA portal and select the correct editing language in the "Language & Resources" area.



Find and replace

Find:

□ Whole words only
☑ Match case
□ Find in substructures
□ Find in hidden texts
☑ Use wildcards
□ Use regular expressions

Languages & resources

Editing language: English (United States)

Reference language: English (United States)

Label

● Text ○ Text list

Off: Text

On □

Unsupported languages

The migration of language-dependent content depends on whether or not WinCC supports the respective language.

If a project only contains project languages not supported by WinCC, the project will not be migrated.

If a project contains supported and unsupported project languages, only the supported languages will be migrated. The editing language and reference language are set to a supported language.

The following languages are not supported by WinCC:

- Arabic
- Hebrew
- Dhivehi
- Gujarati
- Kannada
- Tamil
- Telugu
- Urdu
- Punjabi
- Persian
- Syrian

See also

Supported HMI devices (WinCC flexible) (Page 120)

4.7.2.7 Migrating libraries (WinCC flexible)

Introduction

You need to consider two different cases when migrating from libraries:

1. Migrating a project library
2. Migrating a global library

Migrating a project library

A project library is stored together with the project data in the project file. For this reason, a project library is migrated with the same restrictions as the project data.

Migrating a global library

Global libraries are not supported by the migration. The library objects used in the project will be migrated, however. The library objects are copied when used in the project and then no longer have a connection to the library.

To migrate a global library, you must copy or move the objects contained in the library to the project library. The objects are then included in the migration. In WinCC, you move the migrated objects to a new global library that is created. You can copy or move both individual objects or entire library categories.

See also

Supported HMI devices (WinCC flexible) (Page 120)

4.7.3 Migrating runtime data (WinCC flexible)

4.7.3.1 Migrating runtime data (WinCC flexible)

Introduction

When migrating a project, only the configuration data will be migrated. The runtime data are not affected. You need to update the runtime data following migration.

The runtime data consists of the following:

- Runtime project

  The runtime project contains the compiled project data.

- Recipe data and user administration

  The recipe data and user administration are data that can be changed in runtime.

Migrating runtime data

You update the runtime project by compiling the project in WinCC again and loading it to the HMI device.

If the recipe data and user administration were changed in runtime, you need to back up this information from the HMI device before you load the migrated project. You can then load the migrated project to the HMI device. Finally, you load the saved recipe data and user administration back to the HMI device. You can find additional details on this in the section "Auto-Hotspot".

See also

Migration (WinCC flexible) (Page 114)

Backing up recipe data and user administration (WinCC flexible) (Page 133)

Restoring recipe data and user administration (WinCC flexible) (Page 134)

4.7.3.2 Backing up recipe data and user administration (WinCC flexible)

Introduction

To continue using the recipe data and user administration in a migrated project, you first need to back up this data from the HMI device. Then load the data into the migrated WinCC project. Use ProSave to back up the data.

Requirement

- The WinCC flexible project is running on the HMI device in Runtime.
- The HMI device is connected to a PC on which ProSave is installed.

Procedure

Proceed as follows to back up the recipe data and user administration:

1. Start ProSave.
2. Select the device type and the connection parameters in the "General" tab.
3. Open the "Backup" tab.
4. Select the "Recipes from the device memory" entry in the "Data type" box.
   Do not select "Complete backup" because otherwise you will not be able to select separately when restoring the recipe data.
5. Navigate to the desired location in the "Save as" box and click "Start Backup".
   The recipe data are saved.
6. Select "User administration" in the "Data type" box and click "Start Backup".
   The user administration is saved.

For additional information refer to the online help for ProSave.

Alternative procedure

ProSave is automatically installed with WinCC flexible. The entire functional range of ProSave is available on the configuration PC within WinCC flexible via the menu command "Project > Transfer".

Alternatively, you can back up the recipe data and user administration via the ProSave integrated in WinCC flexible. Start WinCC flexible and select the menu command "Project > Transfer > Backup". Back up the recipe data and user administration as described in steps 4-6.

See also

Migrating runtime data (WinCC flexible) (Page 132)

Restoring recipe data and user administration (WinCC flexible) (Page 134)

4.7.3.3 Restoring recipe data and user administration (WinCC flexible)

Introduction

To continue using saved recipe data and user administration after the migration, you first need to compile the migrated project and load it to the HMI device. You can then transfer the saved data to the HMI device. Use ProSave to restore the data.

Requirement

- The migrated project has been transferred to the HMI device and is running in runtime.
- The HMI device is connected to a PC on which ProSave is installed.

Procedure

Proceed as follows to load the saved recipe data and user administration to the HMI device:

1. Start ProSave.
2. Select the device type and the connection parameters in the "General" tab.
3. Open the "Restore" tab.
4. Navigate to the location of the saved recipe data in the "Opening..." box and select the file.
5. Click "Start Restore".
   The recipe data will be transferred to the HMI device..
6. Repeat steps 4-5 to restore the user administration.
   The user administration will be transferred to the HMI device.

For additional information refer to the online help for ProSave.

Alternative procedure

ProSave is automatically installed with WinCC. The entire functional range of ProSave is available on the configuration PC within WinCC flexible via the menu command "Project > Transfer".

You can also restore the recipe data and user administration via the ProSave integrated in WinCC. Start WinCC and select the menu command "Online > Device maintenance > Restore". Restore the recipe data and user administration as described in steps 4-6.

See also

Migrating runtime data (WinCC flexible) (Page 132)

Backing up recipe data and user administration (WinCC flexible) (Page 133)

4.7.4 Migrating integrated projects (WinCC flexible)

4.7.4.1 Migrating integrated projects (WinCC flexible)

Introduction

The controllers and HMI devices contained in a project integrated in STEP 7 are linked together by the configuration. The configuration data of WinCC flexible and STEP 7 are also connected. When an integrated project is migrated, the complete project will be migrated with components from WinCC flexible and STEP 7. The connections remain intact.

Migrating an integrated project

When migrating an integrated project, the same requirements apply for the WinCC flexible component as those for the migration of a non-integrated WinCC flexible project. The objects and properties contained in the WinCC flexible component must be supported by WinCC, for example, the HMI device or the communication driver. The "Online" property must be activated on the configured connection. A connection with deactivated "Online" property is not migrated.

In addition to the requirements for the WinCC flexible component, there are also requirements for the STEP 7 component of the integrated project. The objects and properties contained in the STEP 7 V5.4 SP5 or V5.5 component must be supported in STEP 7. For detailed information, refer to the documentation for STEP 7.

To fully migrate an integrated project and then edit it, the following components must be installed on the PC performing the migration:

- STEP 7 V5.4 SP5 or STEP 7 V5.5
- WinCC
- STEP 7

If you only want to save the project in migration format, you can use the migration tool. See Basics on migration (WinCC flexible) (Page 115) for additional information.

An integrated project is always fully migrated. If you only want to migrate the WinCC flexible project it contains, you need to separate it from the STEP 7 project before the migration. To separate the project from the integrated form, open the project in STEP 7 V5.4 SP5 or V5.5. Open the WinCC flexible project in the SIMATIC Manager. The project is opened with WinCC flexible. In WinCC flexible, select the menu command "Project > Copy project from STEP 7". WinCC flexible saves a non-integrated copy of the project.

See also

Basics on migration (WinCC flexible) (Page 115)

4.7.4.2 Migrating an integrated project (WinCC flexible)

Introduction

When migrating an integrated project, the components from both the WinCC flexible project and the STEP 7 project will be migrated. This means you need to select the project file with the file extension "*.s7p" for migration. During migration, the data is copied from the existing project and migrated to a new project. You cannot migrate to an existing project.

The migration can be started in both the Portal view and the Project view.

You should only migrate a project in a newly started TIA portal.

If you only want to save the project in migration format, you can use the migration tool. See Basics on migration (WinCC flexible) (Page 115) for additional information.

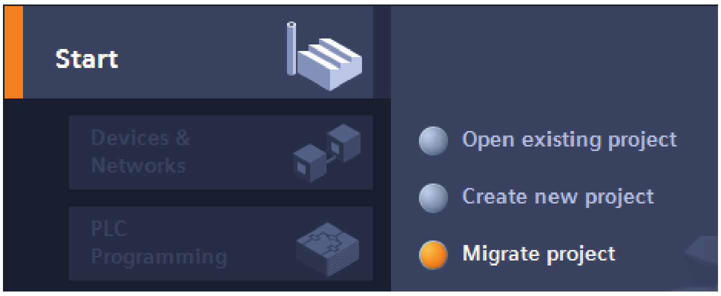
Requirement

- STEP 7 V5.4 SP5 or STEP 7 V5.5 and all add-on packages used are installed.
- STEP 7 and all add-on packages used are installed.
- The TIA portal is newly started.
- No project is open in WinCC.
- An integrated project is available.
- The integrated project is not open.

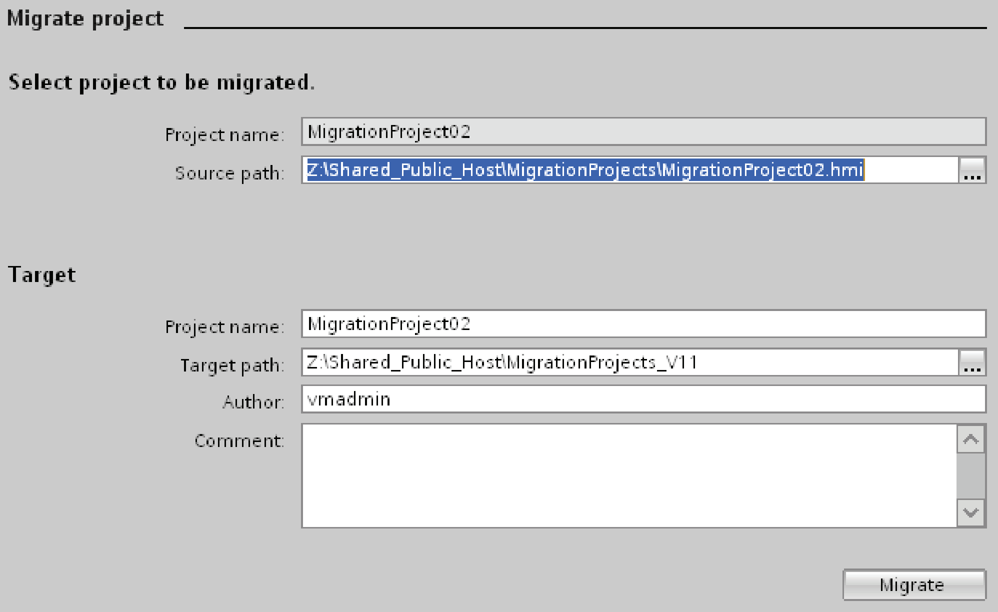
Procedure

Proceed as follows to migrate an integrated project in the Portal view:

1. Select the action "Start > Migrate Project".



2. In the "Source path" box, navigate to the project you want to migrate.



Migrate project

Select project to be migrated.

Project name: MigrationProject02

Source path: Z:\Shared_Public_Host\MigrationProjects\MigrationProject02.hmi

Target

Project name: MigrationProject02

Target path: Z:\Shared_Public_Host\MigrationProjects_V11

Author: vmadmin

Comment:

3. Select the "*.s7p" project file.
4. Change the information for the project to be created, if necessary. For example, change the project name or project path. The data to be migrated is created in the new project.
5. Click "Migrate".
   A new project is created and migration of the data is started:
   - The Project view opens.
   - The progress of the migration is shown in a migration window.
   - Warnings and errors about the migration process are displayed in the Inspector window under "Info > General".
   - All information about the migration is saved in a log file.
   - A message is displayed upon completion of the migration. The message contains a link that you can use to open the log.
6. Once migration is completed, save the project.

Once the migration is complete, you will find a newly created device for each migrated HMI device and controller in the project tree. These devices include the migrated data.

Opening the migration log at a later point in time

The migration log is saved together with the migrated project. You can view the log at a later point in time. Open the log as follows:

1. Select the project in the project tree.
2. Select the "Properties" command in the shortcut menu.
3. Click "Project History" in the "Properties" dialog.
4. Click on the log file.
   The migration log opens.

See also

Basics on migration (WinCC flexible) (Page 115)

4.7.5 Reference (WinCC flexible)

4.7.5.1 Migration of data types (WinCC flexible)

Introduction

To harmonize the data types used by controllers and HMI systems, some types of internal HMI tags are renamed. The naming takes place in accordance with IEC conventions. Because only the names change, there are no changes to the internal tags for the configuration.

The following table describes the mapping of data types from WinCC flexible to the data types in WinCC.

Migration of data types

The internal data types are mapped as follows during migration:

Internal data types WinCC flexible	Internal data types WinCC
Bool	Bool
Char	SInt
Byte	USInt
Int	Int
UInt	UInt
Long	DInt
ULong	UDInt
Float	Real
Double	LReal
String	WString
DateTime	DateTime

Migrating external data types

See the following pages for how to map the available communication drivers.

See also

Migration (WinCC flexible) (Page 114)

Migrating data types of Allen-Bradley DF1 (WinCC flexible) (Page 140)

Migrating data types of Allen-Bradley Ethernet IP (WinCC flexible) (Page 141)

Migrating data types of Mitsubishi FX (WinCC flexible) (Page 142)

Migrating data types of Modicon Modbus (WinCC flexible) (Page 143)

Migrating data types of Modicon Modbus TCP/IP (WinCC flexible) (Page 143)

Migrating data types of Omron Hostlink/Multilink (WinCC flexible) (Page 144)

Migrating data types of SIMATIC S7 200 (WinCC flexible) (Page 145)

Migrating data types of SIMATIC S7 300/400 (WinCC flexible) (Page 146)

4.7.5.2 Migrating data types of Allen-Bradley DF1 (WinCC flexible)

Migrating data types Allen-Bradley DF1

The data types of the Allen-Bradley DF1 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
ASCII	ASCII
BCD4	UInt
BCD8	UDInt
Bit	Bool
Int	Int
Long	DInt
Real	Real
UInt	UInt
ULong	UDInt

See also

Migration of data types (WinCC flexible) (Page 139)

4.7.5.3 Migrating data types of Allen-Bradley Ethernet IP (WinCC flexible)

Migrating data types Allen-Bradley Ethernet IP

The data types of the Allen-Bradley Ethernet IP communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	Bool
DInt	DInt
Int	Int
Real	Real
SInt	SInt
String	String
UDInt	UDInt
UInt	UInt
USInt	USInt

See also

Migration of data types (WinCC flexible) (Page 139)

4.7.5.4 Migrating data types of Mitsubishi FX (WinCC flexible)

Migrating data types Mitsubishi FX

The data types of the Mitsubishi FX communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
12 Bit Block	12-Bit Block
16 Bit Block	16-Bit Block
20 Bit Block	20-Bit Block
24 Bit Block	24-Bit Block
28 Bit Block	28-Bit Block
32 Bit Block	32-Bit Block
4 Bit Block	4-Bit Block
8 Bit Block	8-Bit Block
Bit	Bool
Double	DWord
IEEE-Float	Real
String	String
Word	Word

See also

Migration of data types (WinCC flexible) (Page 139)

4.7.5.5 Migrating data types of Modicon Modbus (WinCC flexible)

Migrating data types Modicon Modbus

The Modicon Modbus communication driver is not supported by WinCC, it is replaced by the Modicon Modbus RTU driver. The data types of the Modicon Modbus communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-Double	+/- Double
+/-Int	+/- Int
16 Bit Group	16 Bit Group
ASCII	ASCII
Bit	Bit
Double	Double
Float	Float
Int	Int

See also

Migration of data types (WinCC flexible) (Page 139)

4.7.5.6 Migrating data types of Modicon Modbus TCP/IP (WinCC flexible)

Migrating data types Modicon Modbus TCP/IP

The data types of the Modicon Modbus TCP/IP communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-Double	+/- Double
+/-Int	+/- Int
16 Bit Group	16 Bit Group
ASCII	ASCII
Bit	Bit
Double	Double
Float	Float
Int	Int

See also

Migration of data types (WinCC flexible) (Page 139)

4.7.5.7 Migrating data types of Omron Hostlink/Multilink (WinCC flexible)

Migrating data types Omron Hostlink/Multilink

The Omron Hostlink/Multilink communication driver is not supported by WinCC, it is replaced by the Omron Host Link driver. The data types of the Omron Hostlink/Multilink communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-DEC	Int
+/-LDEC	DInt
ASCII	String
BIN	Bool
BYTE	Byte
DEC	UInt
IEEE	Real
LDEC	UDInt

See also

Migration of data types (WinCC flexible) (Page 139)

4.7.5.8 Migrating data types of SIMATIC S7 200 (WinCC flexible)

Migrating data types SIMATIC S7 200

The data types of the SIMATIC S7 200 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	Bool
Byte	Byte
Char (Range of values: -128...127)	Char (Range of values: 0...255)
DInt	DInt
DWord	DWord
Int	Int
Real	Real
StringChar	StringChar
Timer	Timer
Word	Word

See also

Migration of data types (WinCC flexible) (Page 139)

4.7.5.9 Migrating data types of SIMATIC S7 300/400 (WinCC flexible)

Migrating data types SIMATIC S7 300/400

The data types of the SIMATIC S7 300/400 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	Bool
Byte	Byte
Char	see below
Counter	see below
Date	Date
Date and Time	Date_And_Time
DInt	DInt
DWord	DWord
Int	Int
Real	Real
String	String
StringChar	see below
Time	Time
Time of Day	Time_Of_Day
Timer	see below
Word	Word

Special considerations for some data types

There are special considerations to be made when migrating external tags that contain data types of a SIMATIC S7-300/400 PLC.

Mapping of the S7 data type "Char"

The S7 data type "Char" is a data type for mapping characters according to the specification. However, since this data type is often used for reading and writing numerical values, it is mapped in WinCC to the S7 data type "Byte". If this should be the case during migration, an alarm will appear in the output window.

If the S7 data type "Char" is used for numerical values and negative numbers were configured at the point of use, the result is an error in mapping to the S7 data type "Byte". The S7 data type "Byte" cannot map any negative numbers. You have to adapt the configuration accordingly to correct the error. Use a signed data type, such as the data type "Int", for processing positive and negative numerical values.

If the S7 data type "Char" is used for mapping characters, you must change the configuration after migration. To represent characters, use the data type "String".

When an integrated project is migrated, the data type "Char" in WinCC is also migrated to the data type "Byte". With a connected PLC tag, the data type "Char" remains "Char". As a result of changing the data type of the HMI tag, symbolic addressing of the tags in question is not migrated. After migration. the tags are interconnected by absolute addresses and continue to work. If you want to restore symbolic addressing, you have to change the configuration accordingly after the migration.

Mapping an array of the S7 data type "Char"

An array of the S7 data type "Char" is mapped to an array of the data type "Byte" during migration.

If an array of the S7 data type "Char" is used for numerical values and negative numbers were configured at the point of use, the result is an error in mapping to an array of the S7 data type "Byte". The S7 data type "Byte" cannot map any negative numbers. You have to adapt the configuration accordingly to correct the error. Use a signed data type, such as the data type "Int", for processing positive and negative numerical values.

Mapping of the S7 data type "Counter"

An external tag with the S7 data type "Counter" with counter address is mapped to the S7 data type "Counter". The address will be retained.

If an external tag with the S7 data type "Counter" addresses a data block or a bit memory address, it is is mapped to the S7 data type "Word". The address will be retained. The migration sets the coding to "SimaticBCDCounter".

The S7 data type "Counter" has a value range of 0-999. When supplied by the S7 data type "Word" the value range may be exceeded on the PLC side. Ensure that you are observing the value range.

Example:

WinCC flexible

Tag	S7 data type	Address	Comment
Counter_Actual_Value	Counter	C10	BCD coded counter value
Counter_Setpoint_Value	Counter	DB10.DBW200	BCD coded counter value
Counter_Setpoint_Value#2	Counter	MW20	BCD coded counter value

WinCC

Tag	S7 data type	Address	Coding	Comment
Counter_Actual_Value	Counter	%C10	<Standard>	BCD coded counter value
Counter_Setpoint_Value	Word	%DB10.%DBW200	SimaticBCDCounter	BCD coded counter value
Counter_Setpoint_Value#2	Word	%MW20	SimaticBCDCounter	BCD coded counter value

Mapping of the data type "StringChar"

In WinCC there is no corresponding data type to which the "StringChar" data type can be mapped. Mapping in WinCC depends on the property "Length" of the S7 data type.

A tag of the "StringChar" data type with the "Length" property > 1 is migrated to an array of the S7 data type "Char". The length of the array corresponds to the length of the originally configured data type "StringChar".

If the property "Length" = 1, the data type in WinCC is migrated to an array of the S7 data type "Char" with length = 1. The expression for an array with an element is "Array[0 ..0] of Char".

Mapping of the S7 data type "Timer"

An external tag with the S7 data type "Timer" with timer address is mapped to the S7 data type "Timer". The address will be retained.

If an external tag with the S7 data type "Timer" addresses a data block or a bit memory address, it is is mapped to the S7 data type "S5 Time". The address will be retained.

Example:

WinCC flexible

Tag	S7 data type	Address	Comment
Timer_Actual_Value	Timer	T10	BCD coded timer value
Timer_Setpoint_Value	Timer	DB10.DBW200	BCD coded timer value
Timer_Setpoint_Value#2	Timer	MW20	BCD coded timer value

WinCC

Tag	S7 data type	Address	Comment
Timer_Actual_Value	Timer	%T10	BCD coded timer value
Timer_Setpoint_Value	S5Time	%DB10.%DBW200	BCD coded timer value
Timer_Setpoint_Value#2	S5Time	%MW20	BCD coded timer value

See also

Migration of data types (WinCC flexible) (Page 139)

4.8 Post-editing integrated projects

4.8.1 Migrating an integrated project

Introduction

When an integrated project is migrated, the complete project will be migrated with components from Wind and STEP 7. Configured connections between control and visualization remain intact.

Migrating an integrated project

When migrating an integrated project, the same requirements apply for the STEP 7 component as those for migration of a non-integrated STEP 7 project. The objects and properties contained in the Wind component must also be supported in Wind (IA Portal). For detailed information, refer to the documentation for Wind.

To fully migrate an integrated project, the following components must be installed on the PH/PC performing the migration:

- STEP 7 V5.4 SP5 or STEP 7 V5.5
- WinCC V7 SP1 or SP2 or Wind Flexible 2008 SP2

To be able to fully post-edit an integrated project, the following components must be installed on the PC for post-editing:

- STEP 7 Professional V11 (IA Portal)
- Wind Basic, Wind Comfort/Advanced oder Wind Professional, depending on the components used

If the initial project is not located on the same PH/PC on which the IA Portal is installed, first use the migration tool to prepare migration. This enables you to convert the initial project to a format that can be read for the IA Portal.

Migration of the STEP 7 part of an integrated project

An integrated project is always fully migrated. Individual components cannot be migrated on their own. You can only migrate the included STEP 7 project alone, if you have previously deleted all HMI stations in the SIMATIC stations in the SIMATIC manager and then recompiled the project in NetPro.

Alternatively, you can open the project in an installation of STEP 7 V5.4 S or V5.5 without an installation of Wind. Then, save the project again and select the "Reorganize" function during saving. The Wind parts are then automatically removed during the saving of the copy.

You can then migrate the STEP 7 project without the Wind project.

Migration of an integrated project without the hardware configuration

You can migrate an integrated project just like any other project without the hardware configuration. In this case, only the software of the initial project is migrated. The STEP 7 devices, network configurations, connections and interrupts used in the initial project are not migrated. However, HMI devices are always retained even if you migrate without hardware configuration.

HMI modules that are plugged into a PC station are converted to a separate Station during the migration. Thus, the migrated project contains a non-specified SIMATIC PC Station and a SIMATIC PC Station with the HMI devices. References to HMI devices are not imported during migration.

The following chapters describe, by way of example, how you post-edit an integrated project after the migration, in order to restore the full functionality of source project.

Storage location of an integrated Wind project

If you migrate an integrated project, the HMI portion of it must be on the same PH/PC as the STEP 7 portion of the project. If the HMI portion is on a different PH, then only the STEP 7 portion will be migrated.

Unsupported objects

The following components are not supported for migration:

- STEP 7 multiproject
  A STEP 7 multiproject cannot be migrated. Migration will be canceled.
- Central Archive Server - CAS
  If a CAS is part of an integrated project, then the migration will be carried out but the CAS data will not be migrated.

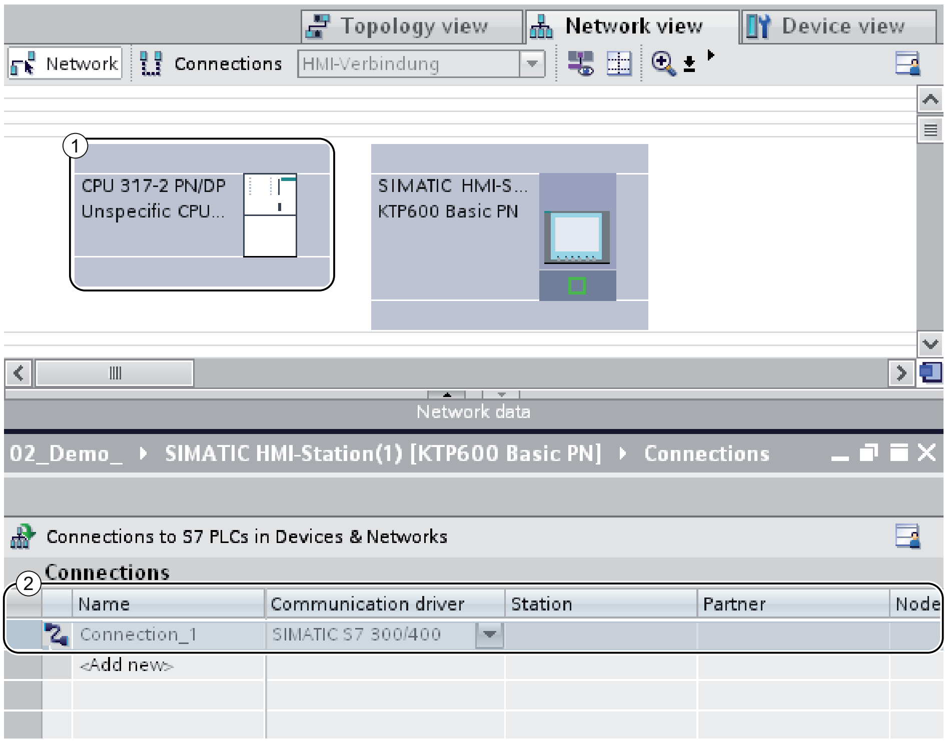
See also

Post-editing integrated projects (Page 152)

4.8.2 Post-editing integrated projects

If you have migrated an integrated project without hardware configuration, unspecified CPUs are used instead of the CPUs of the original project. Since no connection can exist between an unspecified CPU and an HMI device, connections from the source project are also imported only unspecified.

The following figure shows the state after a Migration without hardware configuration in an example project:



Name	Communication driver	Station	Partner	Node
Connection_1	SIMATIC S7 300/400			
<Add new>				

① The original CPU 317-2 PN/DP was replaced with an unspecified CPU during migration.

② The link between the CPU and HMI device is also unspecified and must be renewed.

Procedure

To continue to use an integrated project after the migration, follow these steps:

1. Convert the unspecified devices into suitable devices again.
2. Restore the integrated HMI connection between the HMI device and the PLC.
3. Connect all HMI tags to the newly created integrated connection.
4. Restore the connection between HMI tags and PLC tags.
5. Delete the non-integrated HMI connection.

In the following chapters a sample project is used to describe the individual steps in more detail.

See also

Converting unspecified CPUs into specified CPUs (Page 153)

Creating an integrated HMI connection (Page 155)

Re-linking HMI tags (Page 157)

Deleting an unspecified connection (Page 158)

4.8.3 Converting unspecified CPUs into specified CPUs

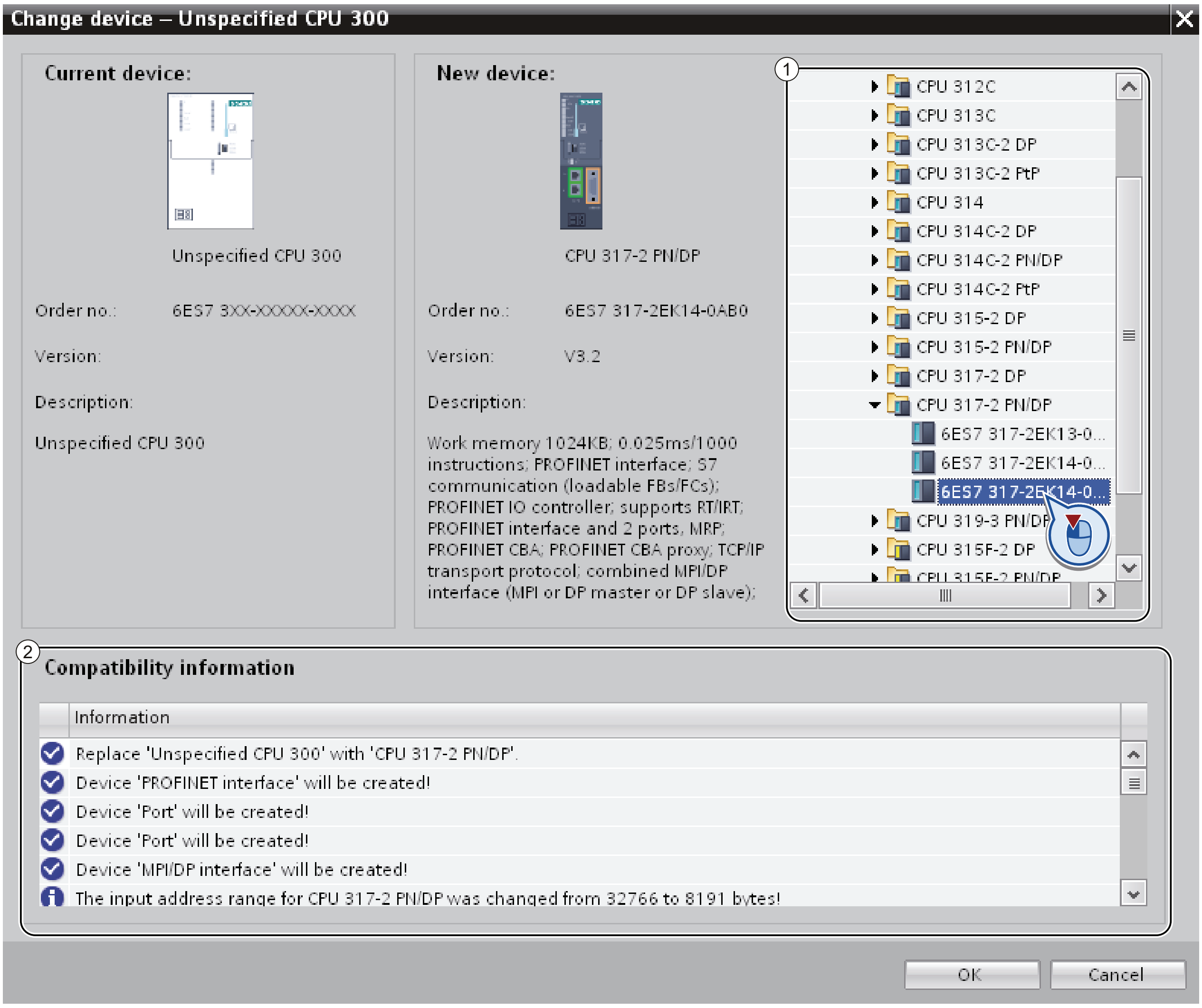
The first step after the migration without hardware configuration is the conversion of the unspecified CPUs into specified CPUs. Unspecified CPUs are placeholders for certain CPUs from the hardware catalog that are not currently known. You can define general parameters and home the CPUs already in the user program. However, the project is not fully functional until the unspecified CPU has been specified.

Specifying a CPU using module replacement

To use module replacement to specify an unspecified CPU, follow these steps:

1. Select the unspecified CPU in the network or device view.
2. Select the "Replace device" command in the shortcut menu.

   The "Replace device" dialog opens.



Change device – Unspecified CPU 300

Current device:

Unspecified CPU 300

Order no.: 6ES7 3XX-XXXXX-XXXX

Version:

Description:

Unspecified CPU 300

New device:

CPU 317-2 PN/DP

Order no.: 6ES7 317-2EK14-0AB0

Version: V3.2

Description:

Work memory 1024KB; 0.025ms/1000 instructions; PROFINET interface; S7 communication (loadable FBs/FCs); PROFINET IO controller; supports RT/IRT; PROFINET interface and 2 ports, MRP; PROFINET CBA; PROFINET CBA proxy; TCP/IP transport protocol; combined MPI/DP interface (MPI or DP master or DP slave);

Compatibility information

Information
Replace 'Unspecified CPU 300' with 'CPU 317-2 PN/DP'.
Device 'PROFINET interface' will be created!
Device 'Port' will be created!
Device 'Port' will be created!
Device 'MPI/DP interface' will be created!
The input address range for CPU 317-2 PN/DP was changed from 32766 to 8191 bytes!

3. Under "New device" in the tree structure, select the module with which you want to replace the unspecified CPU. (Area 1)

   "Compatibility information" provides you with information on the extent to which the selected CPU is compatible with the configuration in source project. (Area 2)
4. Click "OK".
5. Perform the above-described steps for all unspecified CPUs.

See also

Creating an integrated HMI connection (Page 155)

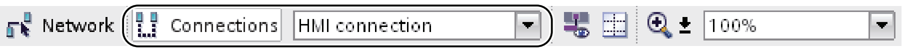
4.8.4 Creating an integrated HMI connection

After you have specified the unspecified CPU, establish the connection to the HMI-device.

Procedure

To create a connection graphically, follow these steps:

1. On the toolbar, click the "Connections" icon. This activates connection mode.



2. Select the connection type "HMI connection" in the adjacent drop-down list.

   The network view highlights in color all CPUs and HMI devices that can be used for an HMI connection.

3. You can now have the connection path automatically determined, or explicitly select a connection path via specific interfaces:

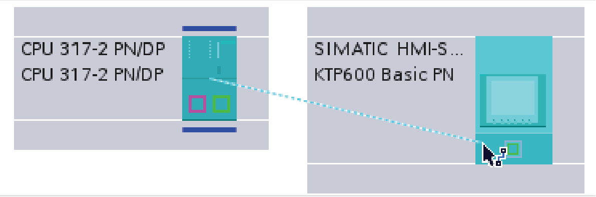
   – Allow connection path to be automatically determined

     Select the source CPU for a connection. Drag the mouse to the target components. Confirm the connection endpoint with another mouse click.

     Alternatively: While holding down the shift button, select the target components and with the right mouse button select the "Add new connection" command.

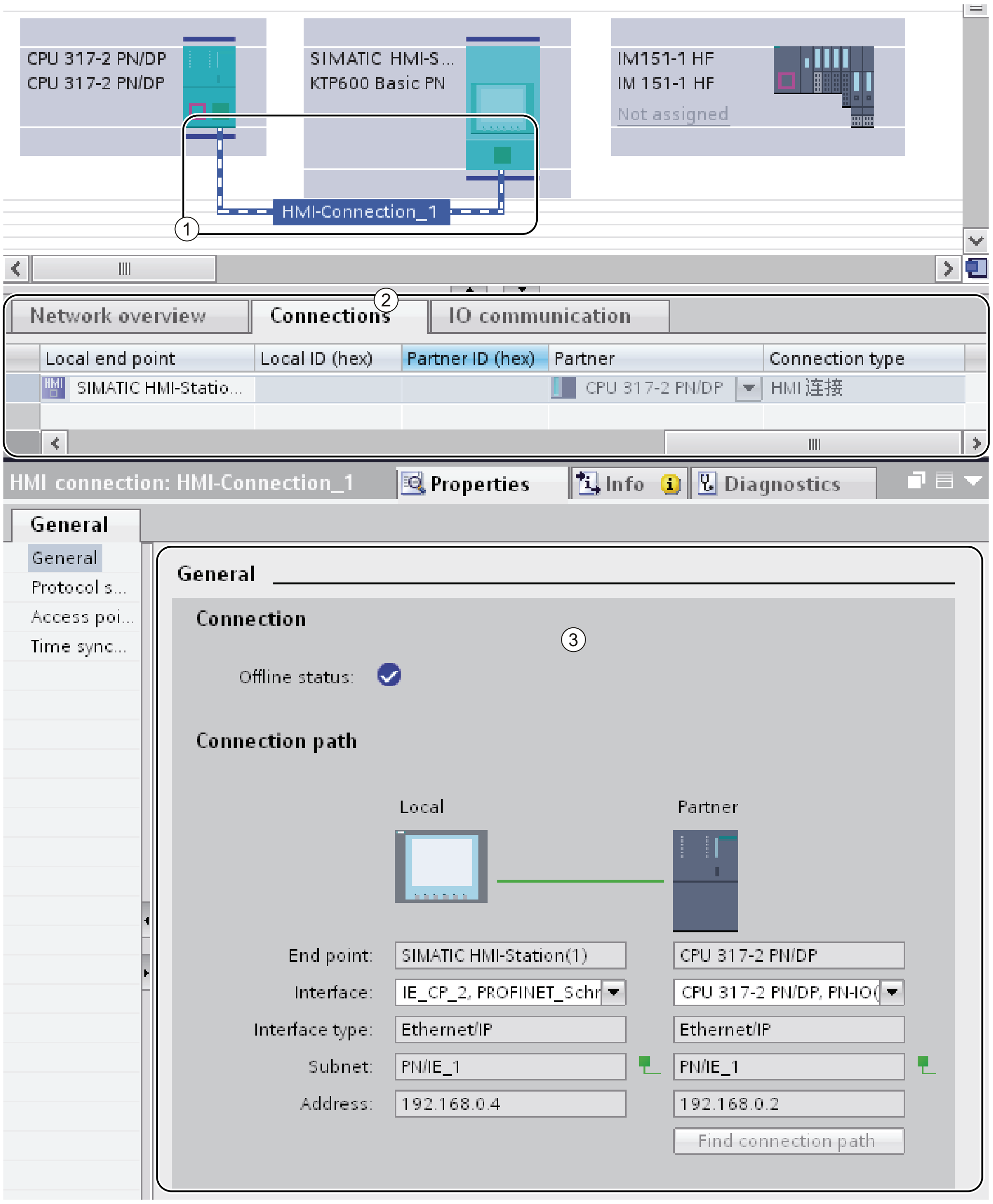
   – Selecting an explicit connection path from interface to interface

     Click on the subnet interface in the device for which you want to create a connection. Hold down the mouse button, drag the cursor to the relevant interface in the target device and then release the mouse button.



Result

The following figure shows the state after the integrated connection has been created:



Local end point	Local ID (hex)	Partner ID (hex)	Partner	Connection type
SIMATIC HMI-Statio...			CPU 317-2 PN/DP	HMI 连接

General

Connection

Offline status: ✓

Connection path

	Local	Partner
End point:	SIMATIC HMI-Station(1)	CPU 317-2 PN/DP
Interface:	IE_CP_2, PROFINET_Schr	CPU 317-2 PN/DP, PN-IO(
Interface type:	Ethernet/IP	Ethernet/IP
Subnet:	PN/IE_1	PN/IE_1
Address:	192.168.0.4	192.168.0.2
		Find connection path

① An integrated HMI connection is created and highlighted in the network view.

② The connection is shown in the connection table of the components.

③ The connection can be edited in the connection properties.

See also

Re-linking HMI tags (Page 157)

4.8.5 Re-linking HMI tags

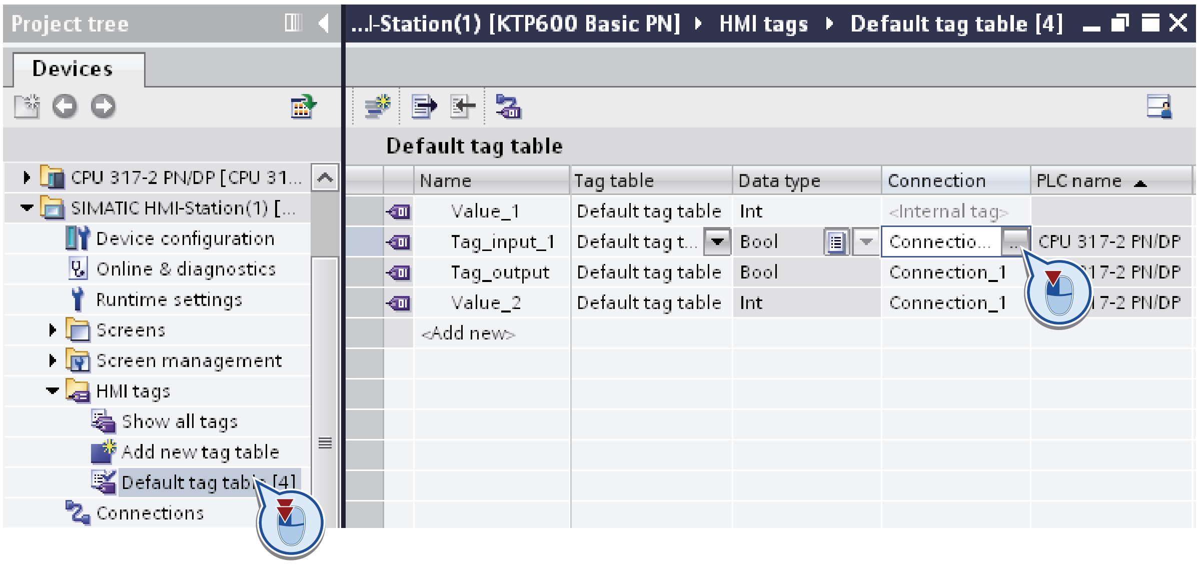
When you have created a new HMI connection between the CPU and HMI device, you have to assign the existing HMI tags to the new connection. Perform the following steps for each line in the relevant tag table.

Procedure

To re-link HMI tags, follow these steps:

1. In the project tree, navigate to the HMI tags and double-click the relevant tag table to show this in the work area.

   The tag table opens.



Project tree

Devices

- CPU 317-2 PN/DP [CPU 31...
- SIMATIC HMI-Station(1) [...
  - Device configuration
  - Online & diagnostics
  - Runtime settings
  - Screens
  - Screen management
  - HMI tags
    - Show all tags
    - Add new tag table
    - Default tag table [4]
  - Connections

...I-Station(1) [KTP600 Basic PN] ▸ HMI tags ▸ Default tag table [4]

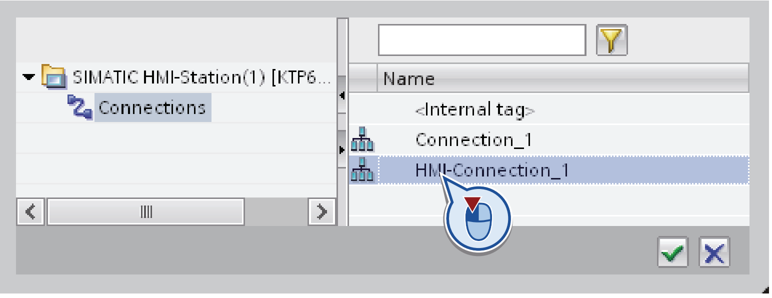
Default tag table

Name	Tag table	Data type	Connection	PLC name
Value_1	Default tag table	Int	<Internal tag>	
Tag_input_1	Default tag t...	Bool	Connectio...	CPU 317-2 PN/DP
Tag_output	Default tag table	Bool	Connection_1	17-2 PN/DP
Value_2	Default tag table	Int	Connection_1	7-2 PN/DP
<Add new>				

2. Click the " ... " button in the "Connection" column.

   A dialog box for selecting the connection opens.

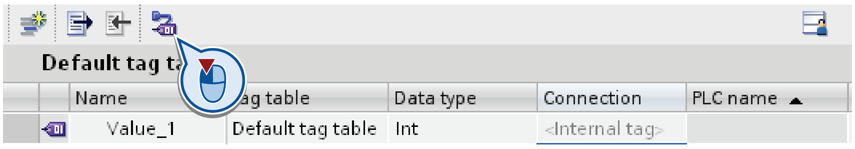
3. Select the newly established HMI connection.



- SIMATIC HMI-Station(1) [KTP6...
  - Connections

Name
<Internal tag>
Connection_1
HMI-Connection_1

4. Click the "✓" button to apply the selected connection.
5. On the toolbar, click the "Re-connect PLC tag" button.



Default tag ta

Name	ag table	Data type	Connection	PLC name ▲
Value_1	Default tag table	Int	<Internal tag>	

See also

Deleting an unspecified connection (Page 158)

4.8.6 Deleting an unspecified connection

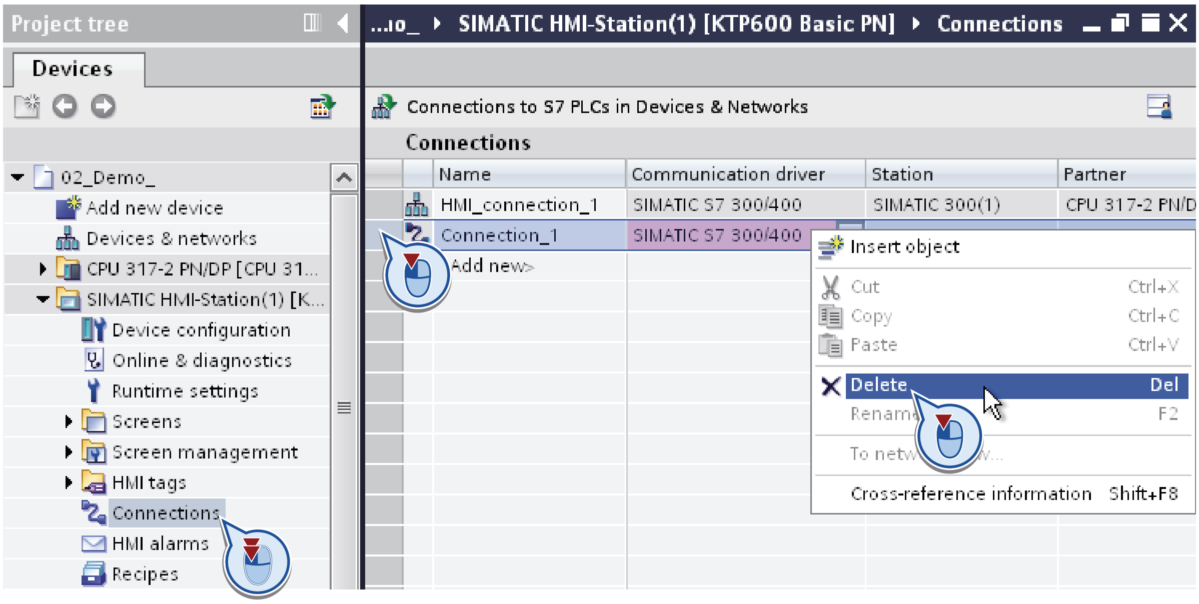
Finally, you can remove unspecified connections that still remain from the source project.

Procedure

To delete unspecified connections, follow these steps:

1. In the project tree, open the HMI device and double-click the "Connections" entry.

   The connection table opens.



Connections

Name	Communication driver	Station	Partner
HMI_connection_1	SIMATIC S7 300/400	SIMATIC 300(1)	CPU 317-2 PN/D
Connection_1	SIMATIC S7 300/400		
<Add new>			

2. Select the row with the old connection in the table.
3. Select the "Delete" command in the shortcut menu of the connection line.
4. Perform the above-described steps for all unspecified connections of the source project.

First steps 5

5.1 Getting Started Documentation

Getting started with the TIA Portal

The Getting Started documentation is available to help you begin using the TIA portal.

The Getting Started documentation contains instructions which show you, step-by-step how to create a project in the TIA portal and give you the chance to get a quick overview of all the possibilities the TIA portal offers you.

Contents

The Getting Started documents describe the creation of a single, continuous project that is extended with each chapter. You start with simple basic functions, and use more complex ones as you continue with the creation of the project.

In addition to the step-by-step instructions, the Getting Started documents also give you background information that explains the functions used and illustrate how they relate to each other.

Target audience

The Getting Started documents are intended for beginners, but are also useful for users migrating from a previous version of SIMATIC STEP 7 and WinCC.

Download

The documentation is available, free of charge at the Service&Support (https://support.automation.siemens.com) portal in PDF form.

You can download the documents here:

- STEP 7 Basic (http://support.automation.siemens.com/WW/view/en/40263542/0/en)
- STEP 7 Professional (http://support.automation.siemens.com/WW/view/en/28919804/133300)

Introduction to the TIA Portal 6

6.1 User interface and operation

6.1.1 Starting, setting and exiting the TIA Portal

6.1.1.1 Starting and exiting the TIA Portal

Starting the TIA Portal

Follow the steps below to start the TIA Portal:

1. In Windows, select "Start > Programs > Siemens Automation > TIA Portal V11".

   The TIA Portal opens with the last settings used.

Exiting the TIA Portal

Follow the steps below to exit the TIA Portal:

1. In the "Project" menu, select the "Exit" command.

   If the project contains any changes that have not been saved, you will be asked if you wish to save them.

   – Select "Yes" to save the changes in the current project and close the TIA Portal.

   – Select "No" to close the TIA Portal without saving the most recent changes in the project.

   – Select "Cancel" to cancel the closing procedure. The TIA Portal will remain open if you select this option.

6.1.1.2 Overview of the program settings

Overview

The following table shows the application settings that you can make:

Group	Setting	Description
General settings	User name	The user name of the user. The user name is stored in the project properties when a new project is created.
	User interface language	Language for the program interface
	Mnemonic	Specifies the mnemonics for programming: "German" uses the German mnemonics, for example, "E1.0". "International" uses international mnemonics, for example, "I1.0". For information on the differences in the mnemonics of the individual commands, refer to the description of the relevant programming language.
	Show list of recently used projects	Number of entries in the list of recently used projects in the "Project" menu
	Show all message windows	All message windows whose appearance was manually suppressed are displayed again.
	Open cascade automatically in tooltips	After a brief time, the tooltips automatically expand to display a cascade containing additional help. If this option is cleared, the tooltips must be manually expanded.
Layout	Load most recent project during startup	The last opened project is opened automatically after the TIA Portal starts.
	Reset layout	Resets the complete layout of the application to factory settings.
Start view	Most recent view	Starts the program in the last view that was used. This can be be either the portal view or the project view.
	Portal view	The TIA Portal will always be started in the portal view, irrespective of the last view you worked in.
	Project view	The TIA Portal will always be started in the project view, irrespective of the last view you worked in.
View for objects in overview	Details	When several views are available, the detail view opens by default, for example, in the overview window.
	List	When several views are available, the list view opens by default, for example, in the overview window.
	Thumbnails	When several views are available, the symbol view opens by default, for example, in the overview window.
Storage settings	Recently used storage location	When a project is saved the first time, the most recently used file path is set by default.
	Default storage location	Enables the specification of file paths for: • Projects • Libraries

Group	Setting	Description
Data exchange	Storage location for data import	Imported files are searched for at this storage path by default.
	Storage location for data export	This is the default storage path for the data export.
	Storage location for support packages	When support packages are downloaded, they are stored in the specified storage path and can be installed from there.
	Storage location for log files	Log files are stored at the location specified here.

See also

Starting and exiting the TIA Portal (Page 161)

Resetting the user interface layout (Page 193)

Changing the settings (Page 166)

6.1.1.3 Overview of the script and text editor settings

Overview

The following table shows the available settings for script and text editors:

Group	Setting	Description
Font	Font type and size	Sets the font type and size for the text in text editors.
Font colors	Color settings	You can choose the colors for individual text elements from the respective drop-down lists in the text editors. Optional settings are available for the following text elements: • Text • Keywords • Comments • Translatable comments • Instructions • Scripts • Standard functions • System functions • Constant strings • Constant tags • Tags • Object models • Formal parameters
	Reset to default	Resets all font colors in editors to their factory settings.
Tabs	Tab width	Sets the width of tabs.
	Use tabs	Enables the use of tabs.
	Use spaces	Specifies use of space characters instead of tabs.
Indent	Indention at start of paragraph	Specifies whether the start of a new paragraph is to be indented. The following selection options are available: • None No indentation is used at the beginning of a paragraph in editors. • Block The first line of a paragraph in the editors is automatically indented. • Smart The program code is detected and the paragraphs are automatically indented to improve readability of the syntax.
View	Show line numbers	Displays the line numbers on the left side of the text.
	Show white spaces	Shows control characters within a text.

Group	Setting	Description
STL (statement list)	Font type and size	Sets the font type and size for STL program code.
SCL (Structured Control Language)	Font type and size	Sets the font type and size for SCL.
	Indentation	Creates SCL programs automatically with syntactically correct indentation.
	Show line numbers	Shows the row numbers in SCL programs.

See also

Changing the settings (Page 166)

6.1.1.4 Overview of the print settings

Overview

The following table shows the available settings for printing:

Group	Setting	Description
General	Always print table data as pairs of values	Tables are printed as a list and not in tabular form. The corresponding values are listed for each column. Enable this option, for example, if you want to print a table that is too large for the print area.
	Always print data in tables	All parameters of technology objects are printed in tabular format.
	Print mask graphics if possible	If the utilized editor supports this function, the contents of the editor are printed not only as a table but rather as a complete graphic as it appears on the screen.
Hardware configuration	Active graphic view	The graphics of network and device view are included in the printout.
	Active table	A table associated with an editor is included when printing with the editor.
PLC Programming	Zoom factor	Specifies the size in which blocks are to be printed out.
	With interface	The interfaces of blocks are included in the printout.
	With comments	Comments on blocks are included in the printout.
	With line numbers	The line numbers of the program code are printed for text-based programming languages.
HMI screens	Show tab order	In the printout you can specify the order in which the runtime objects can be selected with the TAB key.

See also

Changing the settings (Page 166)

6.1.1.5 Changing the settings

Procedure

To change the settings, proceed as follows:

1. Select the "Settings" command in the "Options" menu.

   The "Settings" window is displayed in the work area.
2. Select the "General" group in the area navigation to change the settings described in the previous sections. Or click on one of the other entries in area navigation to make settings for your installed products.
3. Change the settings.

Result

The change will be adopted immediately, there is no need to save it explicitly.

See also

Overview of the program settings (Page 162)

Overview of the script and text editor settings (Page 164)

Overview of the print settings (Page 165)

6.1.2 Layout of the user interface

6.1.2.1 Views

Views

Two different views are available for your automation project:

- The portal view is a task-oriented view of the project tasks.
- The project view is a view of the components of the project, as well as the relevant work areas and editors.

You can change over between the two views using a link.

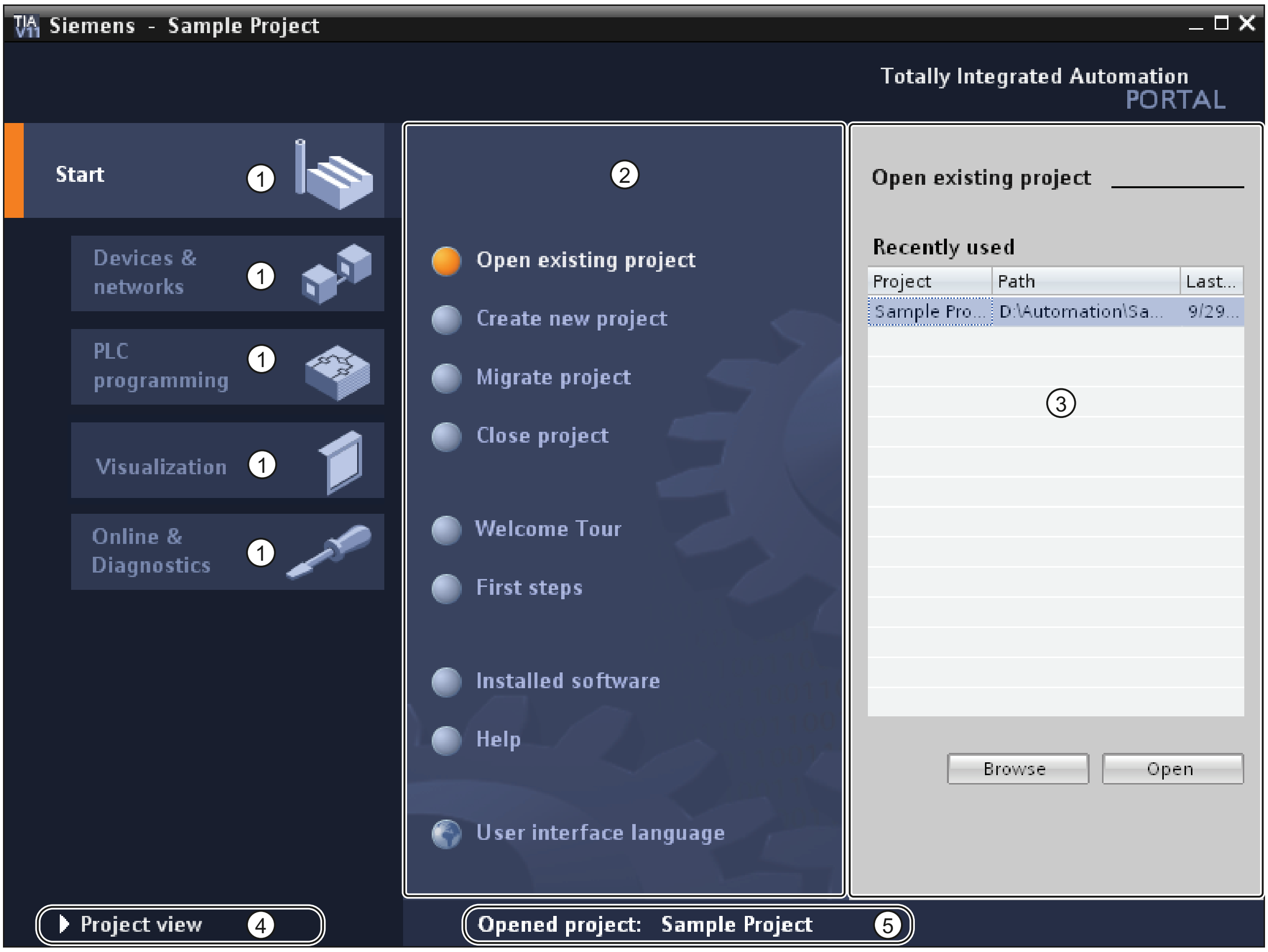
6.1.2.2 Portal view

Purpose of the portal view

The portal view provides you with a task-oriented view of the tools. Here, you can quickly decide what you want to do and call up the tool for the task in hand. If necessary, the view changes automatically to the project view (Page 169) for the selected task.

Layout of the portal view

The following figure shows an example of the components in the portal view:



Project	Path	Last...
Sample Pro...	D:\Automation\Sa...	9/29...

① Portals for different tasks

② Actions for the selected portal

③ Selection panel for the selected action

④ Change to the project view

⑤ Display of the project that is currently open

Portals

The portals provide the basic functions for the individual task areas. The portals that are provided in the portal view depends on the products that have been installed.

Actions for the selected portal

Here, you will find the actions available to you in the portal you have selected. You can call up the help function in every portal on a context-sensitive basis.

Selection panel for the selected action

The selection panel is available in all portals. The content of the panel adapts to your current selection.

Change to the project view

You can use the "Project view" link to change to the project view.

Display of the project that is currently open

Here, you can obtain information about which project is currently open.

See also

Project navigation (Page 171)

Basics of the work area (Page 175)

Inspector window (Page 184)

Basics on task cards (Page 186)

Details view (Page 189)

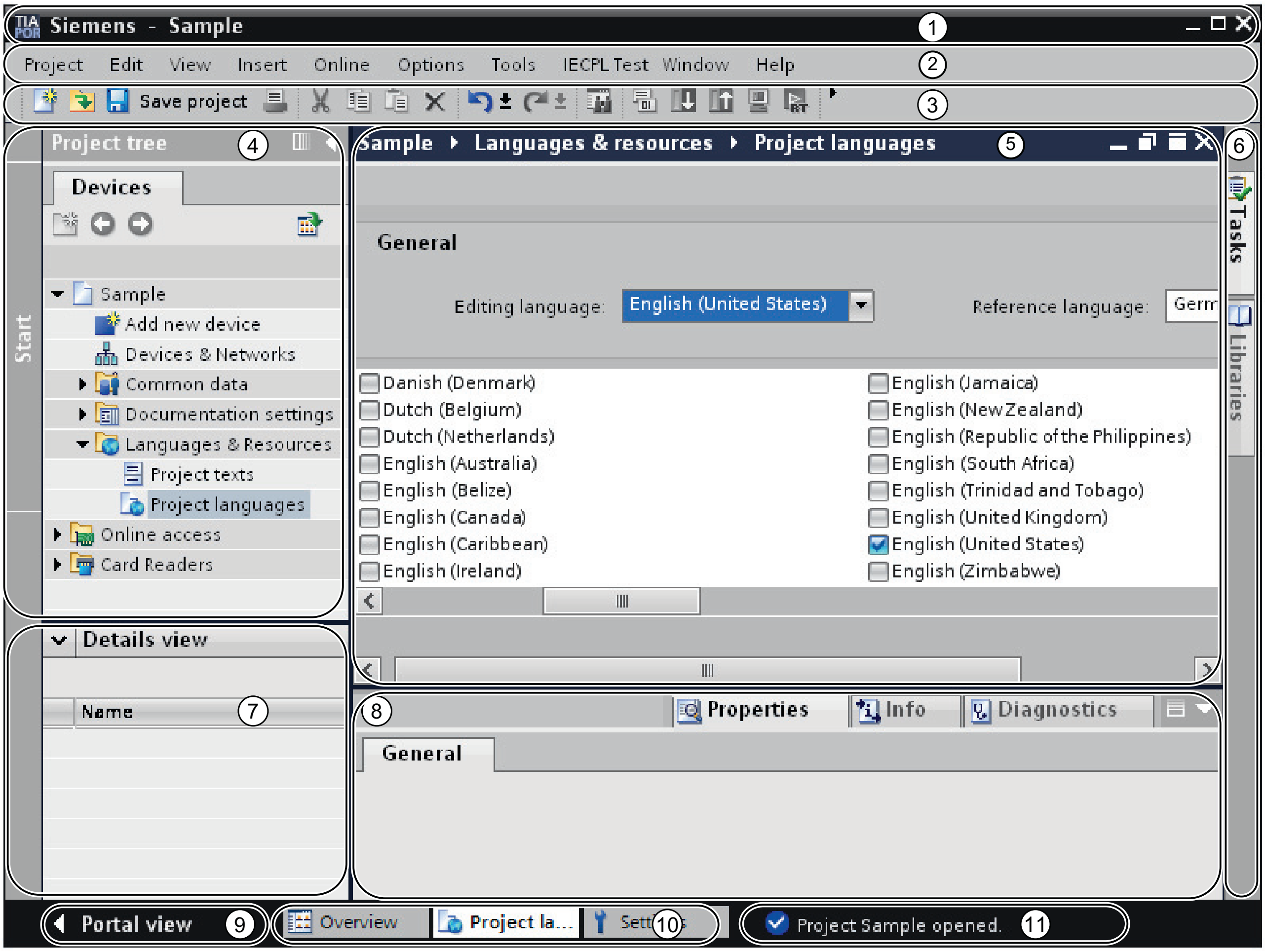
6.1.2.3 Project view

Purpose of the project view

The project view is a structured view of all components of the project.

Layout of the project view

The following figure shows an example of the components of the project view:



Editing language: English (United States)

Reference language: Germ

☐ Danish (Denmark)
☐ Dutch (Belgium)
☐ Dutch (Netherlands)
☐ English (Australia)
☐ English (Belize)
☐ English (Canada)
☐ English (Caribbean)
☐ English (Ireland)
☐ English (Jamaica)
☐ English (New Zealand)
☐ English (Republic of the Philippines)
☐ English (South Africa)
☐ English (Trinidad and Tobago)
☐ English (United Kingdom)
☑ English (United States)
☐ English (Zimbabwe)

①	Title bar
②	Menu bar
③	Toolbar
④	Project tree (Page 171)
⑤	Work area (Page 175)
⑥	Task cards (Page 186)
⑦	Details view (Page 189)
⑧	Inspector window (Page 184)
⑨	Changing to the Portal view (Page 167)
⑩	Editor bar
⑪	Status bar with progress display

Title bar

The name of the project is displayed in the title bar.

Menu bar

The menu bar contains all the commands that you require for your work.

Toolbar

The toolbar provides you with buttons for commands you will use frequently. This gives you faster access to these commands.

Changing to the portal view

You can use the "Portal view" link to change to the portal view.

Editor bar

The Editor bar displays the open editors. If you have opened a lot of editors, they are shown grouped together. You can use the Editor bar to change quickly between the open elements.

Status bar with progress display

In the status bar, you will find the progress display for processes that are currently running in the background. This also includes a progress bar that shows the progress graphically. Hover the mouse pointer over the progress bar to display a tooltip providing additional information on the active background process. You can cancel the background processes by clicking the button next to the progress bar.

If no background processes are currently running, the status bar displays the last generated alarm.

6.1.2.4 Project navigation

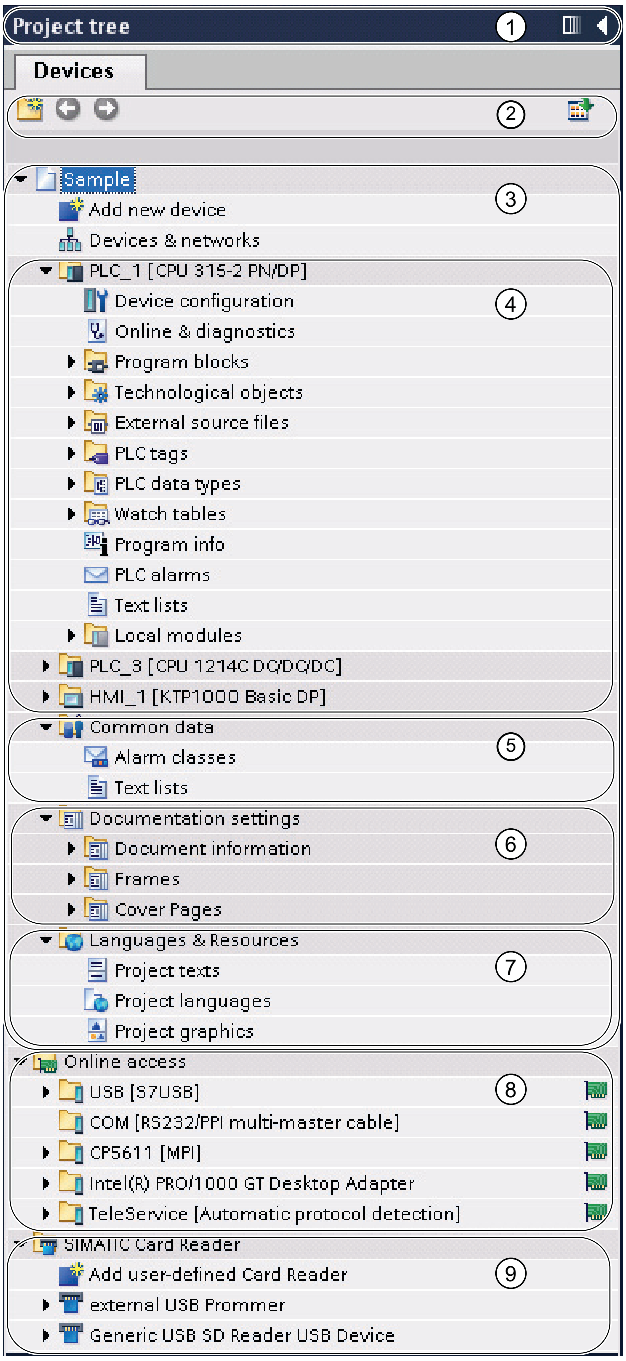
Function of the project tree

Using the project tree features gives you access to all components and project data. You can perform the following tasks in the project tree:

- Add new components
- Edit existing components
- Scan and modify the properties of existing components

Layout of project tree

The following figure shows an example of the project tree components:



①	Title bar
②	Toolbar
③	Project
④	Devices
⑤	Common data
⑥	Document information
⑦	Languages & resources
⑧	Online access
⑨	SIMATIC Card Reader

Title bar

The title bar of the project tree has a button for automatically and manually collapsing the project tree. Once it is collapsed manually, the button is "Reduced" to the left-hand margin. It changes from an arrow pointing left to one that is pointing right, and can now be used to reopen the project tree. You can use the "Reduce automatically" button collapse to project tree automatically when you do not need it.

See also: Maximizing and minimizing the work area (Page 177)

Toolbar

You can do the following tasks in the toolbar of the project tree:

- Create a new user folder; for example, in order to group blocks in the "Program blocks" folder.
- Navigate forward to the source of a link and back to the link itself.

  There are two buttons for links in the project tree. You can use these to navigate from the link to the source and back.
- Show an overview of the selected object in the work area.

  When the overview is displayed, the lower-level objects and actions of the elements in the project tree are hidden.

Project

You will find all the objects and actions related to the project in the "Project" folder, e.g.:

- Devices
- Languages & resources
- Online access

Device

There is a separate folder for each device in the project, which has an internal project name. Objects and actions belonging to the device are arranged inside this folder.

Common data

This folder contains data that you can use across more than one device, such as common message classes, scripts and text lists.

Document information

In this folder, you can specify the layout for project documentation to be printed at a later point.

Languages & resources

You can determine the project languages and texts in this folder.

Online access

This folder contains all the interfaces of the programming device / PC, even if they are not used for communication with a module.

SIMATIC Card Reader

This folder is used to manage all card readers connected to the programming device / PC.

See also

Portal view (Page 167)

Project view (Page 169)

Basics of the work area (Page 175)

Inspector window (Page 184)

Basics on task cards (Page 186)

Details view (Page 189)

6.1.2.5 Work area

Basics of the work area

Function of the work area

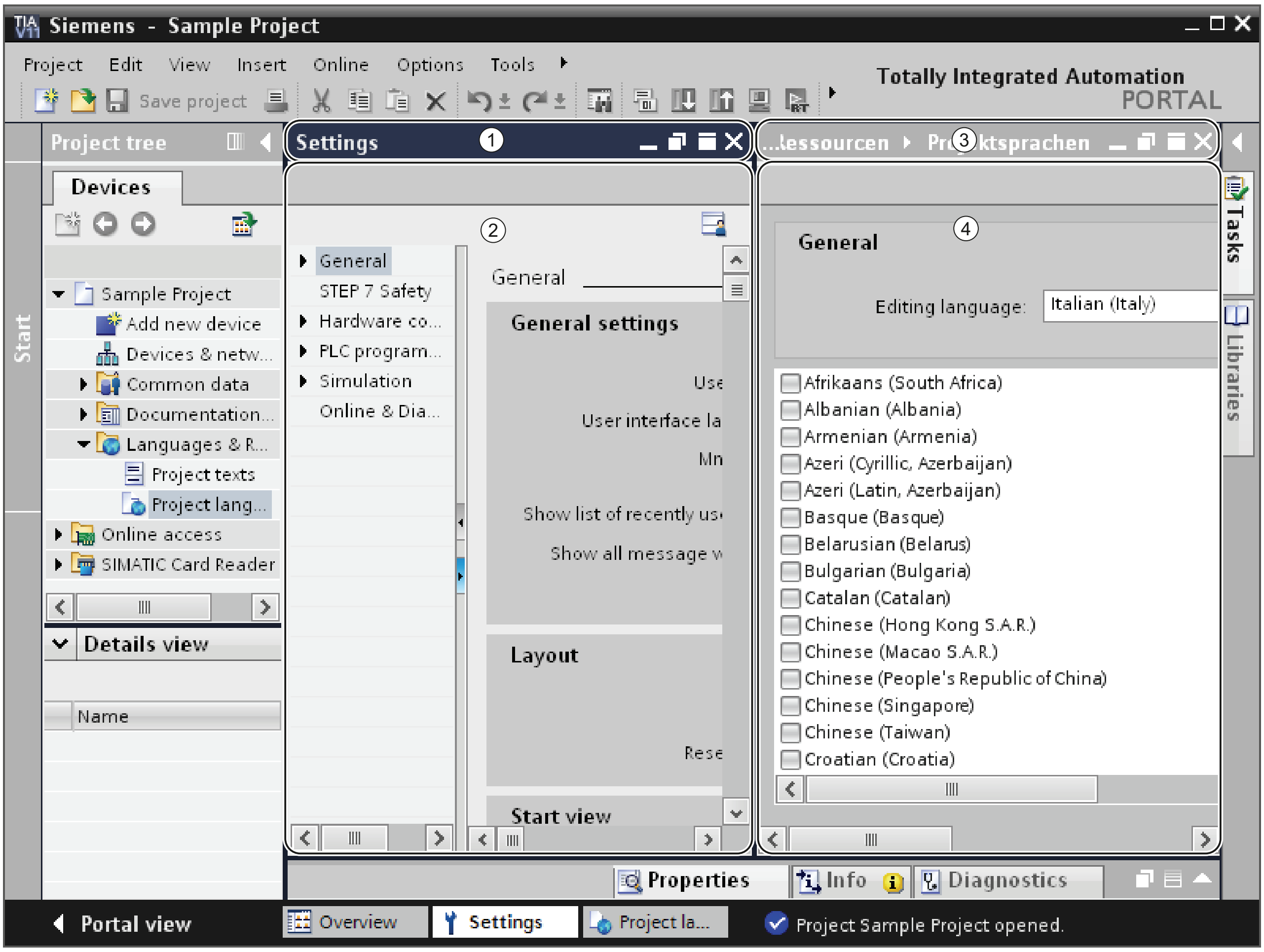
The objects that you can open for editing purposes are displayed in the work area. These objects include, for example:

- Editors and views
- Tables

You can open several objects. However, normally it is only possible to see one of these at a time in the work area. All other objects are displayed as tabs in the Editor bar. If, you would like to view two objects at the same time when performing certain tasks, you can tile the work area either horizontally or vertically or undock elements of the work area. If no objects are open, the work area will be empty.

Layout of the work area

The following figure shows an example of a vertically split work area:



Editing language: Italian (Italy)

Project Sample Project opened.

① Title bar of left-hand editor

② Work area of left-hand editor

③ Title bar of right-hand editor

④ Work area of right-hand editor

See also

Maximizing and minimizing the work area (Page 177)

Splitting the work area (Page 178)

Floating the work area elements (Page 179)

Using grouped elements of the work area (Page 180)

Minimizing and maximizing elements of the work area (Page 181)

Switching between the elements in the work area (Page 182)

Saving a layout of editors and tables (Page 183)

Maximizing and minimizing the work area

You have the option to adapt the work area to make it as large as possible. You can use the following function for this:

- Maximizing the work area

  You can close the task cards, project tree and inspector window with a single click. This increases the size of the work area. You can minimize the work area again at any time in order to return to the previous view.
- Collapsing task cards, project tree, and Inspector window automatically

  You can use the "Collapse automatically" option for the task cards, project tree, and Inspector window. This function causes these items to collapse automatically when you don't need them.

Maximizing and minimizing the work area

To maximize the work area, follow these steps:

1. Open an element such as an editor or a table.

   The element appears in the work area.
2. Click the "Maximize" button in the title bar of the element.

   The task cards, project tree and inspector window collapse, and the work area is shown with its maximum dimensions.

To minimize the work area again, follow these steps:

1. Click the "Embed" button in the title bar of the displayed element.

   This restores the view that existed before the work area was maximized. That is, if the task cards, project tree, or Inspector window were expanded before, they will be expanded again.

Collapsing task cards, project tree, and Inspector window automatically

To collapse the task cards automatically, follow these steps:

1. Click "Collapse automatically" in the title bar of the task cards.

   The task cards collapse when you click anywhere outside the task cards.
2. To use the task cards, click the collapsed task cards.
3. The task cards expand and are available for use. The "Collapse automatically" option remains enabled.

To collapse the project tree automatically, follow these steps:

1. Click "Collapse automatically" in the title bar of the project tree.

   The project tree collapses when you click anywhere outside the project tree.
2. To use the project tree, click the collapsed project tree.

   The project tree expands and is available for use. The "Collapse automatically" option remains enabled.

To collapse the Inspector window automatically, follow these steps:

1. Click "Collapse automatically" in the title bar of the Inspector window.

   The Inspector window collapses when you click anywhere outside the Inspector window.
2. To use the Inspector window, click the collapsed Inspector window.

   The Inspector window expands and is available for use. The "Collapse automatically" option remains enabled.

To disable the automatic collapse option, follow these steps:

1. Click "Expand permanently" again in the relevant window.

   The Collapse automatically" option is disabled, and the window remains expanded.

See also

Basics of the work area (Page 175)

Splitting the work area (Page 178)

Floating the work area elements (Page 179)

Using grouped elements of the work area (Page 180)

Minimizing and maximizing elements of the work area (Page 181)

Switching between the elements in the work area (Page 182)

Saving a layout of editors and tables (Page 183)

Splitting the work area

You can split the work area vertically or horizontally.

Procedure

To split the work area vertically or horizontally, follow these steps:

1. In the "Window" menu, select the "Split editor space vertically" or "Split editor space horizontally" command.

   The element you have clicked and the next element in the Editor bar will be displayed either next to one another or one above the other.

Note

If no elements are open in the work area, the "Split editor space vertically" and "Split editor space horizontally" functions will not be available.

See also

Basics of the work area (Page 175)

Maximizing and minimizing the work area (Page 177)

Floating the work area elements (Page 179)

Using grouped elements of the work area (Page 180)

Minimizing and maximizing elements of the work area (Page 181)

Switching between the elements in the work area (Page 182)

Saving a layout of editors and tables (Page 183)

Floating the work area elements

You can float work area elements in their own separate window:

- Editors
- Tables
- Setting windows
- Task cards
- Inspector window

You can embed floating elements again in the work area at any time.

Floating the work area elements

To float work area elements, follow these steps:

1. Click the "Float" button in the title bar of the element.

   The element will be released from the work area and displayed in its own window. You can now place the window wherever you wish. If you have minimized the window, you can restore it via the editor bar.

Embedding elements in the work area

To embed elements in the work area again, follow these steps:

1. Click the "Embed" button in the title bar of the element.

   The element will appear in the work area again.

See also

Basics of the work area (Page 175)

Maximizing and minimizing the work area (Page 177)

Splitting the work area (Page 178)

Using grouped elements of the work area (Page 180)

Minimizing and maximizing elements of the work area (Page 181)

Switching between the elements in the work area (Page 182)

Saving a layout of editors and tables (Page 183)

Using grouped elements of the work area

If you open more than five elements of the same type, e.g., editors or tables, they are grouped in the editor bar. You can use these groups as follows:

- Displaying individual elements of a group
- Displaying all elements of a group in separate windows
- Embedding all displayed elements of a group in the work area
- Minimizing all displayed elements
- Closing all elements of a group

Displaying individual elements of a group

To display individual elements of a group, follow these steps:

1. In the editor bar, click the group containing the element you want to display.

   All list of all available elements of the group is displayed.

2. Click the element that you want to display.

Displaying all elements of a group in separate windows

To display all elements of a group in separate windows, follow these steps:

1. In the editor bar, right-click the group whose elements you want to display.
2. Select "Restore group" in the shortcut menu.

   All elements of the group are displayed in separate, overlapping windows. Move the windows in order to see the individual element, or choose an element via the group in the editor bar.

Embedding all displayed elements of a group in the work area

To embed all elements of a group displayed in separate windows in the work area again, follow these steps:

1. In the editor bar, right-click the group whose elements you want to embed.
2. Select "Embed group" in the shortcut menu.

   All elements of the group are embedded in the work area again.

Minimizing all displayed elements

To minimize all elements of a group, follow these steps:

1. In the editor bar, right-click the group whose elements you want to minimize.
2. Select "Minimize group" in the shortcut menu.

   All elements of the group are minimized. However, the minimized elements remain open and can be quickly maximized again via the group in the editor bar.

Closing all elements of a group

To close all elements of a group, follow these steps:

1. In the editor bar, right-click the group whose elements you want to close.
2. Select "Close group" in the shortcut menu.

   All elements of the group are closed. The group is removed.

See also

Basics of the work area (Page 175)

Maximizing and minimizing the work area (Page 177)

Splitting the work area (Page 178)

Floating the work area elements (Page 179)

Minimizing and maximizing elements of the work area (Page 181)

Switching between the elements in the work area (Page 182)

Saving a layout of editors and tables (Page 183)

Minimizing and maximizing elements of the work area

You can minimize the elements that are open in the work area, such as editors or tables, as needed. However, an element remains open even if it has been minimized, and can quickly be maximized again using the editor bar.

Minimizing elements in the work area

To minimize elements in the work area, follow these steps:

1. Click the "Minimize" button in the title bar of the element.

   The element is minimized, but can still be accessed via the editor bar.

To minimize all elements at the same time, follow these steps:

1. In the "Window" menu, select the "Minimize all" command.

Maximizing elements in the work area

To maximize elements in the work area again, follow these steps:

1. Click the required element in the editor bar.

   The element is maximized and appears in the work area.

See also

Basics of the work area (Page 175)

Maximizing and minimizing the work area (Page 177)

Splitting the work area (Page 178)

Floating the work area elements (Page 179)

Using grouped elements of the work area (Page 180)

Switching between the elements in the work area (Page 182)

Saving a layout of editors and tables (Page 183)

Switching between the elements in the work area

You can switch between the elements in the work area at any time.

Switching between the elements in the work area

To switch to the previous or next editor, follow these steps:

1. In the "Window" menu, select the "Next editor" or "Previous editor" command.

   The next or previous editor will be displayed.

See also

Basics of the work area (Page 175)

Maximizing and minimizing the work area (Page 177)

Splitting the work area (Page 178)

Floating the work area elements (Page 179)

Using grouped elements of the work area (Page 180)

Minimizing and maximizing elements of the work area (Page 181)

Saving a layout of editors and tables (Page 183)

Saving a layout of editors and tables

You have the option of adapting editors and tables to meet your requirements. For example, you can hide columns in tables that you don't need. You can then save your customized view.

Procedure

To save the layout of editors and tables in the work area, follow these steps:

1. Adapt the editor or table according to your requirements.
2. Click the "Remember Layout" button in the editor or table.

Result

The layout is saved. When you reopen the editor or table, this layout will be used.

See also

Basics of the work area (Page 175)

Maximizing and minimizing the work area (Page 177)

Splitting the work area (Page 178)

Floating the work area elements (Page 179)

Using grouped elements of the work area (Page 180)

Minimizing and maximizing elements of the work area (Page 181)

Switching between the elements in the work area (Page 182)

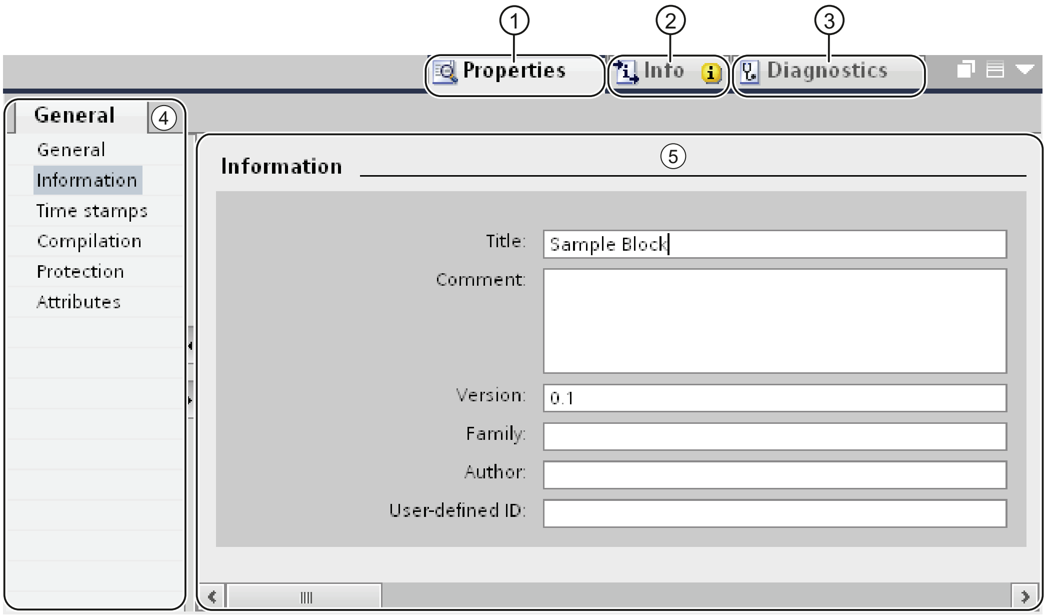
6.1.2.6 Inspector window

Function of the Inspector window

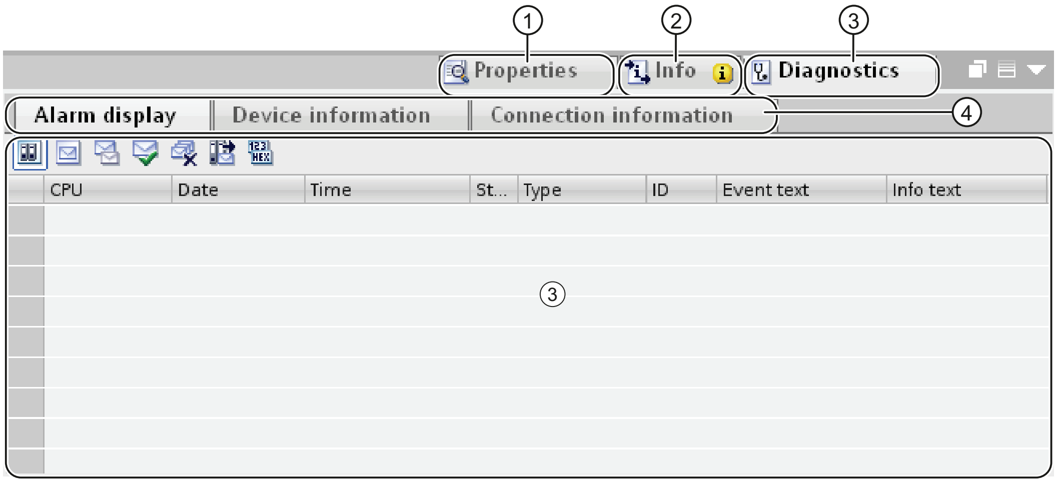
Additional information on an object selected or on actions executed are displayed in the inspector window.

Layout of the Inspector window

The following figures show the components of the Inspector window:



Title: Sample Block
Comment:
Version: 0.1
Family:
Author:
User-defined ID:



CPU	Date	Time	St...	Type	ID	Event text	Info text

① "Properties" tab

② "Info" tab

③ "Diagnostics" tab

④ Navigation within the tabs:

- Area navigation within the "Properties" tab
- Lower-level tabs in the "Info" and "Diagnostics" tabs

"Properties" tab

This tab displays the properties of the object selected. You can change editable properties here.

"Info" tab

This tab displays additional information on the object selected, as well as alarms on the actions executed (such as compiling).

"Diagnostics" tab

This tab provides information on system diagnostics events, configured alarm events, and connection diagnostics.

Navigation within the tabs

You can use area navigation and the lower-level tabs to display the information you require within the tabs.

See also

Project navigation (Page 171)

Basics of the work area (Page 175)

Portal view (Page 167)

Project view (Page 169)

Basics on task cards (Page 186)

Details view (Page 189)

6.1.2.7 Task cards

Basics on task cards

Function of task cards

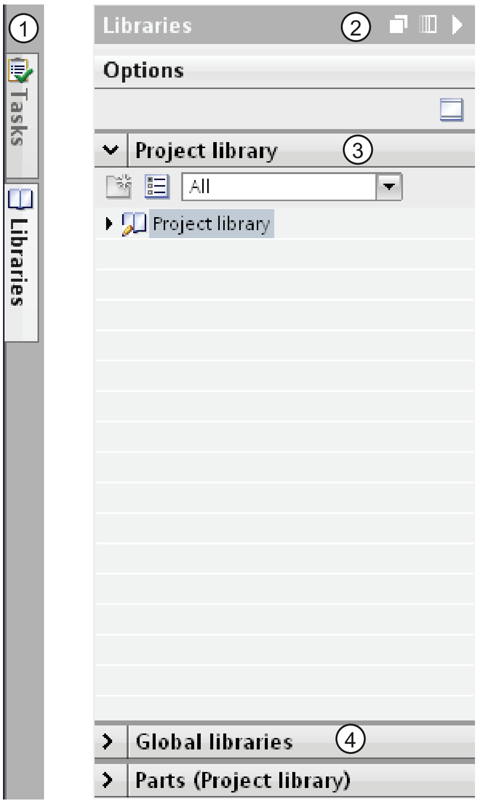
Depending on the edited or selected object, task cards that allow you perform additional actions are available. These actions include:

- Selecting objects from a library or from the hardware catalog
- Searching for and replacing objects in the project
- Dragging predefined objects to the work area

The task cards available can be found in a bar on the right-hand side of the screen. You can collapse and reopen them at any time. Which task cards are available depends on the products installed. More complex task cards are divided into panes that you can also collapse and reopen.

Layout of task cards

The following figure shows an example of the bar with the task cards:



① Task cards closed

② "Libraries" task card open

③ "Project library" pane open

④ "Global libraries" pane closed

See also

Changing the pane mode (Page 188)

Project navigation (Page 171)

Basics of the work area (Page 175)

Inspector window (Page 184)

Portal view (Page 167)

Project view (Page 169)

Details view (Page 189)

Changing the pane mode

You can choose between two pane modes:

- Single pane mode:

  Only one pane is open at any given time. If you open another pane, the previously opened pane is closed automatically.
- Multi-pane mode:

  You can open several panes at the same time.

Procedure

To change the pane mode, follow these steps:

1. Click the "Change pane mode" button above the panes inside a task card.

See also

Basics on task cards (Page 186)

6.1.2.8 Details view

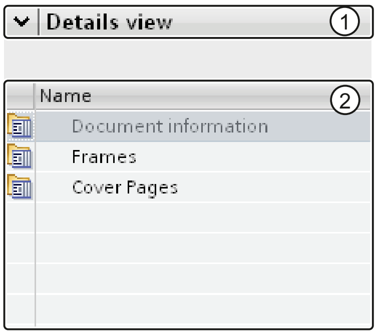
Purpose of the details view

The detail view shows certain content of the selected object is in the overview window or in the project tree. This might include text lists or tags.

The content of folders is not shown, however. To display the content of folders, use the project tree or the Inspector window.

Layout of the details view

The following figure shows an example of the details view:



Name
Document information
Frames
Cover Pages

① Title bar

② Content of the selected object

Title bar

The arrow for closing the details view is located in the title bar of the details view. After it has closed, the direction in which the arrow is pointing changes from left to right. It can now be used to reopen the details view.

Objects

The displayed content varies depending on the selected object. You can move the content of objects from the details view to the required location using drag-and-drop.

See also

Project navigation (Page 171)

Basics of the work area (Page 175)

Inspector window (Page 184)

Basics on task cards (Page 186)

Portal view (Page 167)

Project view (Page 169)

6.1.2.9 Overview window

Overview window

Functions of the Overview window

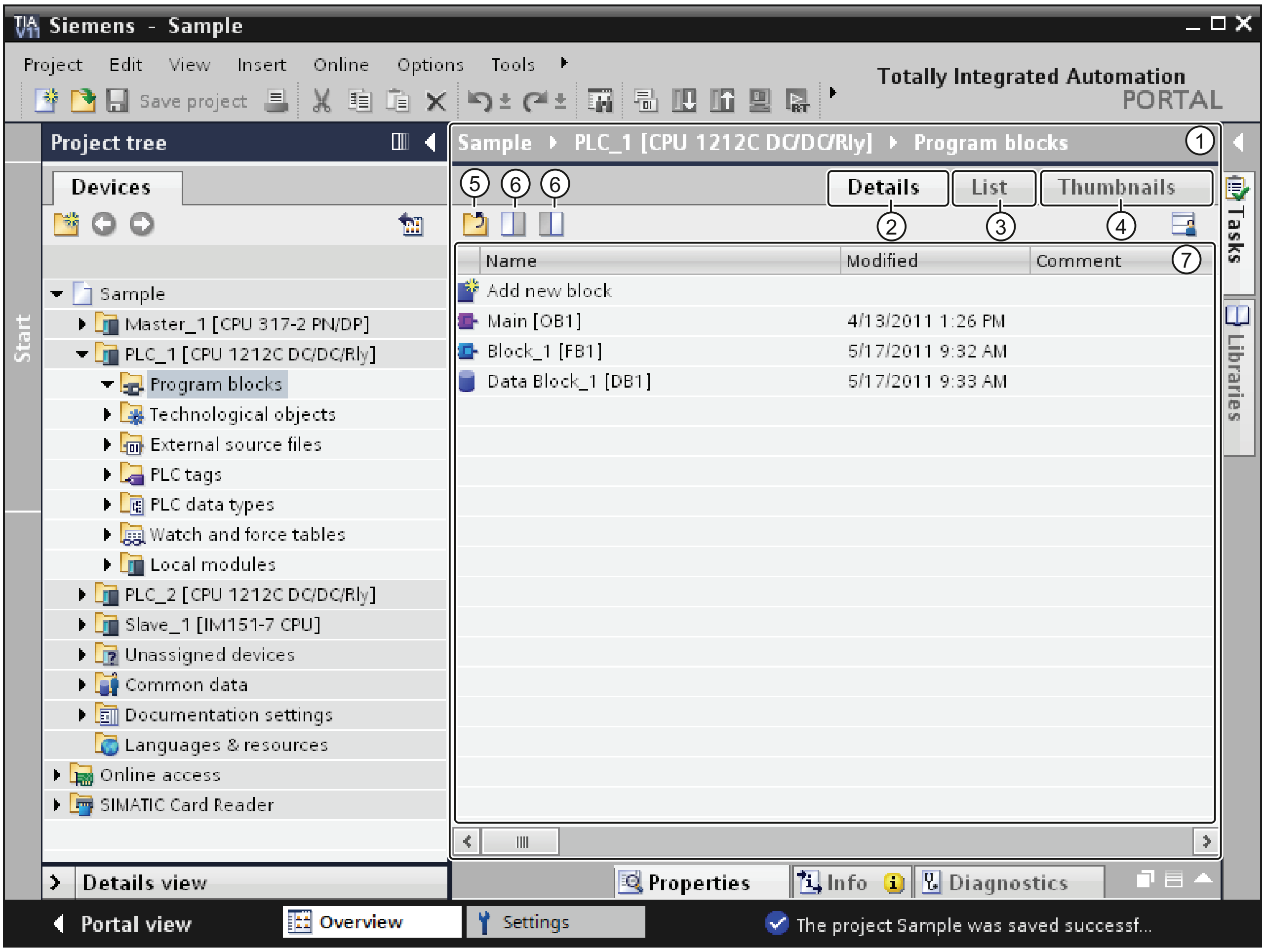
The Overview window supplements the project tree. The Overview window shows the contents of the folder currently selected in the project tree.

In addition, you can perform the following actions in the Overview window:

- Open objects
- Display and edit the properties of objects in the Inspector window
- Rename objects
- Call object-specific actions from the shortcut menu
- Compare objects side by side
- Perform various object operations, such as inserting objects from the library via drag-and-drop and moving, copying, pasting, and deleting objects

Layout of the Overview window

The following figure shows the components of the Overview window:



Name	Modified	Comment
Add new block		
Main [OB1]	4/13/2011 1:26 PM	
Block_1 [FB1]	5/17/2011 9:32 AM	
Data Block_1 [DB1]	5/17/2011 9:33 AM	

The project Sample was saved successf...

① Overview window
② Switch to the Details view
③ Switch to the List view
④ Switch to the Icon view
⑤ Move to higher level
⑥ Split the overview window in two. Either the right or left half of the overview window is synchronized. Clicking again cancels the split.
⑦ Contents of the object selected in the project tree.

Display forms of the Overview window

The content of the Overview window can be displayed as follows:

- Details view

  The objects are displayed in a list with additional information, such as the date of the last change.

- List view

  The objects are displayed in a simple list.

- Icon view

  The objects are displayed as icons.

See also

Comparing objects (Page 192)

Showing or hiding additional columns (Page 192)

Comparing objects

You can display the contents of two folders or objects side by side in the Overview window. The Overview window is split in half and you can display different contents on the left and right sides.

In addition, you can use a drag-and-drop operation to move objects between the split windows. Thus, for example, you can move contents from one window to the other.

Procedure

To split the Overview window in half or cancel the split, follow these steps:

1. In the toolbar, click on the "Synchronize left side" or "Synchronize right side" icon to split the overview window. Either the left or the right side of the overview window synchronized with the contents of the selected object in the project tree.
2. To cancel the split, click again on the previously selected icon.

See also

Overview window (Page 190)

Showing or hiding additional columns

In the details view of the Overview window, you can display more columns containing additional information on an object and then hide them again. The columns available depend on the selected object.

Procedure

To show or hide additional table columns, follow these steps:

1. Right-click the title bar of the table.
2. Select the "Show/Hide" command in the shortcut menu, and select the columns you want to display.

See also

Overview window (Page 190)

6.1.2.10 Resetting the user interface layout

Every change you make to the layout of the user interface is saved. The changes are thus available even after a restart of the TIA Portal. For example, if you change the height and width of a text editor or the division of a table, your changes are retained so that you don't have to re-customize elements every time.

In some cases, however, it may be helpful to restore the original layout settings, for example, to suit the preferences of another user or if a different user interface layout is desired.

Procedure

To reset the user interface settings to the factory settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.

   The "Settings" window is displayed in the work area.
2. Select the "General" group in the area navigation.
3. Click the "Reset to default" button under "Layout > Reset layout".

Result

The default settings for the user interface are restored.

See also

Overview of the program settings (Page 162)

6.1.3 Keyboard shortcuts

6.1.3.1 Keyboard shortcuts for project editing

Editing a project

Function	Key combination	Menu command
Open a project	<Ctrl+O>	Project > Open
Close a project	<Ctrl+W>	Project > Close
Save a project	<Ctrl+S>	Project > Save
Save a project under a different name	<Ctrl+Shift+S>	Project > Save as
Print project	<Ctrl+P>	Project > Print
Compile a project	<Ctrl+B>	Edit > Compile
Undo last action	<Ctrl+Z>	Edit > Undo
Redo last action	<Ctrl+Y>	Edit > Redo

Editing objects within a project

Function	Key combination	Menu command
Rename a project	<F2>	Edit > Rename
Highlight all objects in an area	<Ctrl+A>	Edit > Select all
Copy an object	<Ctrl+C>	Edit > Copy
Cut an object	<Ctrl+X>	Edit > Cut
Paste an object	<Ctrl+V>	Edit > Paste
Delete an object	<Del>	Edit > Delete
Find an object	<Ctrl+F>	Edit > Find and replace
Replace an object	<Ctrl+H>	-

Calling up the help function

Function	Key combination	Menu command
Calling up the help function	<F1> or <Shift+F1>	Help > Show help

6.1.3.2 Keyboard shortcuts for windows

Opening and closing windows

Function	Key combination	Menu command
Open/close project tree	<Ctrl+1>	View > Project tree
Opening/closing the detailed view	<Ctrl+4>	View > Details view
Opening/closing the overview	<Ctrl+2>	View > Overview
Opening/closing a task card	<Ctrl+3>	View > Task card
Open/close inspector window	<Ctrl+5>	View > Inspector window
Close all editors	<Ctrl+Shift+F4>	Window > Close all
Open the shortcut menu	<Shift+F10>	-

6.1.3.3 Keyboard shortcuts in the project tree

Keyboard shortcuts in the project tree

Function	Key combination
Jump to the start of the project tree	<Home> or <Page Up>
Jump to the end of the project tree	<End> or <Page Down>
Open folder	<Arrow Right>
Close folder	<Arrow Left>

6.1.3.4 Keyboard shortcuts in tables

General keyboard shortcuts in tables

Function	Key combination
Place a cell in edit mode	<F2> or <Return>
Open drop-down list in a cell	<Return>
Close drop-down list in a cell	<Esc>

Navigate in tables

Function	Key combination
Go to the next cell	<Arrow keys>
Go to the next editable cell on the right	<Tab>
Go to the next editable cell on the left	<Shift+Tab>

Function	Key combination
Move a screen upwards	<PgUp>
Move a screen downwards	<PgDn>
Go to the first cell in the row	<Home>
Go to the last cell in the row	<End>
Go to the first cell in the table	<Ctrl+Home>
Go to the last cell in the table	<Ctrl+End>
Go to the top cell in the column	<Ctrl+up arrow>
Go to the bottom cell in the column	<Ctrl+down arrow>

Highlighting areas in tables

Function	Key combination
Highlight a column	<Ctrl+space bar>
Highlight a row	<Shift+space bar>
Highlight all cells	<Ctrl+A>
Highlight to expand a cell	<Shift+arrow keys>
Extend highlighting to the first visible cell	<Shift+PgUp>
Extend highlighting to the last visible cell	<Shift+PgDn>
Extend highlighting to the first row	<Ctrl+Shift+up arrow>
Extend highlighting to the last row	<Ctrl+Shift+down arrow>
Extend highlighting to the first cell in the row	<Ctrl+Shift+left arrow>
Extend highlighting to the last cell in the row	<Ctrl+Shift+right arrow>

6.1.3.5 Keyboard shortcuts for text editing

Editing text

Function	Key combination
Switch to insert or overwrite mode	<Insert>
Exit edit mode	<Esc>
Delete	<Del>
Delete characters	<Backspace>

6.1.3.6 Using the on-screen keyboard

Introduction

When working with the TIA portal, you also have the Microsoft on-screen keyboard available.

Displaying the on-screen keyboard

To display the on-screen keyboard, follow these steps:

1. In the "View" menu, select the "Screen keyboard" command.

Exiting the on-screen keyboard

To exit the on-screen keyboard, follow these steps:

1. In the "File" menu of the on-screen keyboard, select the "Exit" command.

6.1.4 Special features specific to the operating system

6.1.4.1 Influence of user rights

Restrictions when user rights are limited

The software provides several functions that require direct access to the hardware of the programming device / PC and therefore also to the installed operating system. To make full use of the range of functions, the software must cooperate closely with the operating system. To ensure problem-free interaction, you should therefore be logged on to the operating system with adequate user rights.

In particular, you may not be able to use functions requiring an online connection or those that change the settings of interface cards if you work with limited user rights.

Recognizing restricted functions

You can recognize functions requiring special rights as follows:

- A shield icon is displayed beside the function.

  🛡

  The function can be used but is regulated by the user account control.
- A box is grayed out and cannot be accessed.

  You require administrator privileges to access the box. In some operating system environments, you can obtain administrator privileges by entering an administrator password.

Note

A box being grayed out does not necessarily mean a lack of rights. You should also check the additional information in the tooltip cascades to find out the conditions for editing the box.

6.1.4.2 Expanding user rights

Counteracting restrictions due to user rights

Certain functions may not be available if you are not logged on to the operating system with adequate rights. You can counteract these restrictions in the following ways:

- Enabling of extended rights using Windows user account control
- Logging on to the operating system with administrator privileges
- Using temporary administrator rights

Enabling extended rights using the Windows user account control

To be able to use a function indicated by the shield icon of the Windows user account control, follow these steps:

1. Click on the box or button with the shield icon.

   The security prompt of the Windows user account control opens.
2. Follow the instructions of the Windows user account control and, when prompted enter an administrator password, if possible.

The function can now be used once without restrictions.

Logging on to the operating system with administrator privileges

To be able to use a function that is disabled due to lack of user rights, follow these steps:

1. Close the software.
2. Log off from the operating system.
3. Log on to the operating system with administrator privileges.
4. Restart the software.

Using temporary administrator rights

To obtain administrator privileges temporarily, follow these steps:

1. Click the "Change settings" button. You will find this button in dialogs that allow the temporary assignment of administrator privileges.

   An operating system dialog box for entering an administrator password opens.
2. Enter an administrator password.

The settings can be temporarily changed. When you call the dialog again, the procedure must be repeated.

Note

This function is not supported by all operating systems. If no "Change settings" button is present or the button is grayed out, you will need to log on to the operating system with administrator privileges instead.

6.2 Help on the information system

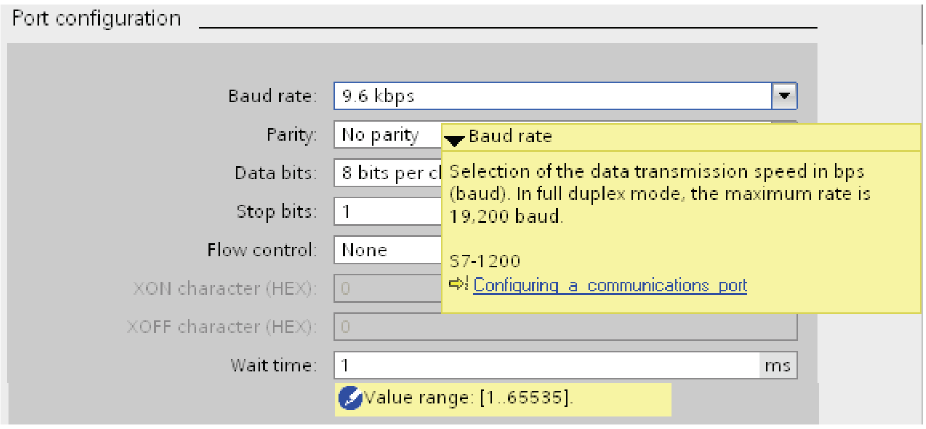
6.2.1 General remarks on the information system

Quick answers to your questions

A comprehensive Help system is available for solving your tasks. It describes basic concepts, instructions and functions. While working with the program, you also receive the following support:

- Roll-out for correct inputs in dialog boxes
- Tooltips for information on elements of the user interface, for example text boxes, buttons and icons. Some of the tooltips are supplemented by cascades containing more precise information.
- Help on the current context, on menu commands for example when you click on the keys <F1> or <Shift+F1>.

The following figure shows an example of a cascading tooltip (top) and a roll-out (bottom):



Port configuration

Baud rate: 9.6 kbps
Parity: No parity
Data bits: 8 bits per c
Stop bits: 1
Flow control: None
XON character (HEX): 0
XOFF character (HEX): 0
Wait time: 1 ms

Baud rate
Selection of the data transmission speed in bps (baud). In full duplex mode, the maximum rate is 19,200 baud.

S7-1200
Configuring a communications port

Value range: [1..65535].

Help

The Help system describes concepts, instructions and functions. It also contains reference information and examples. The help opens in a separate window.

A navigation pane appears on the left side of the help window. You can also hide the navigation pane to make room on the screen. The navigation pane provides you with the following functions:

- Table of contents
- Search in the index
- Full text search of the entire Help
- Favorites

Identification of the topics in the Help according to the type of information

The help topics are identified by different symbols depending on the type of information they contain.

Symbol	Information type	Explanation
⇨ 1. 2.	Operating instructions	Describes the steps to follow in order to carry out a particular task.
▶▶	Example	Contains a concrete example to explain the task.
▶▤	Factual information	Contains background information that you need to know to carry out a task.
↗	Reference	Contains comprehensive reference information to refer back to.

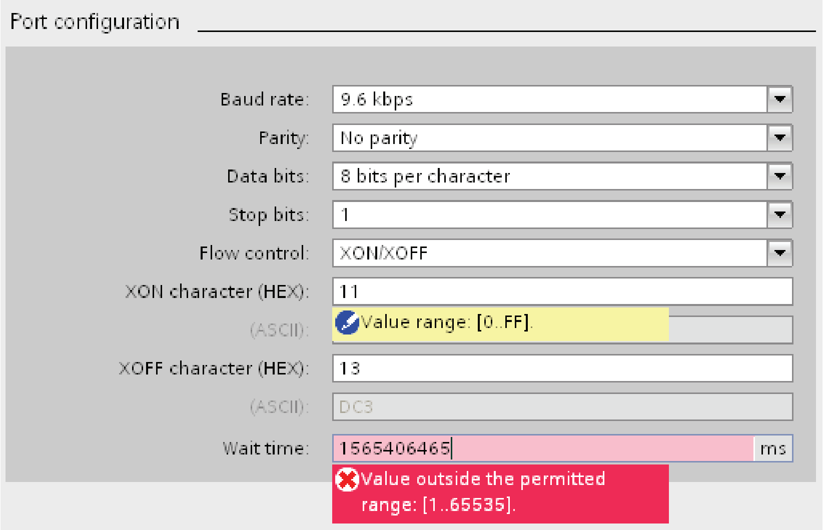
Identification of the topics in the Help according to the target system

Depending on the products that are installed, the help system may contain sections that apply only to specific devices. To be able to recognize such sections at a glance, you will find a note in brackets in the table of contents. The search results in the full text search and in the index are marked in the same way if they only apply to a specific device.

Roll-out

Certain text boxes offer information that rolls out and helps you to enter valid parameters and values. The roll-out informs you about permissible value ranges and data types of the text boxes.

The following figure shows a roll-out (yellow) and a roll-out error message (red), which indicates an invalid value:



Port configuration

Baud rate: 9.6 kbps

Parity: No parity

Data bits: 8 bits per character

Stop bits: 1

Flow control: XON/XOFF

XON character (HEX): 11

(ASCII): Value range: [0..FF].

XOFF character (HEX): 13

(ASCII): DC3

Wait time: 1565406465 ms

Value outside the permitted range: [1..65535].

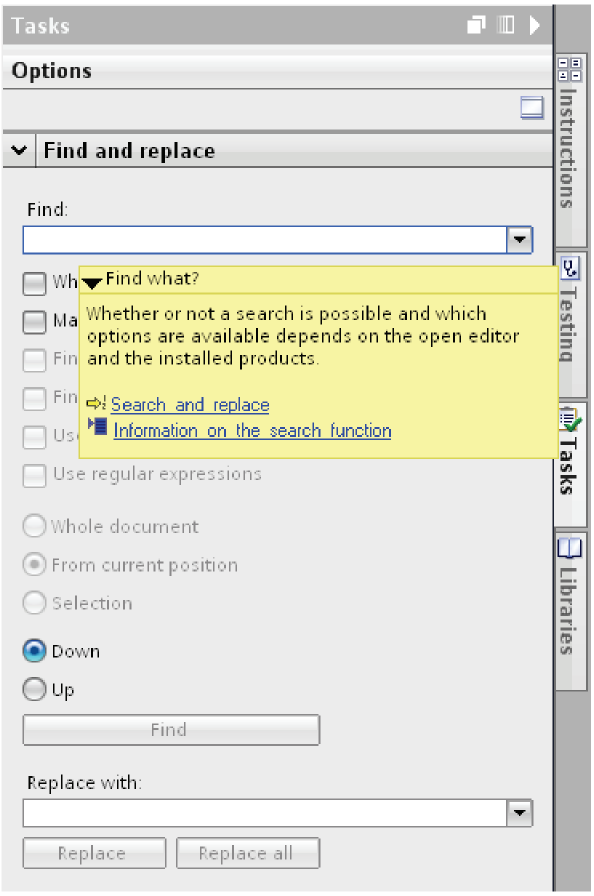
Tooltip

Interface elements offer you a tooltip for easier identification.

Tooltips, which have an arrow icon on the left, contain additional information in tooltip cascades. If you position the mouse pointer briefly over the tooltip or click the arrow icon, this information is displayed. The automatic display of tooltip cascades can be disabled.

If additional information is contained in the Help system, a link appears to the corresponding Help topic in the cascade. If you click on the link, the corresponding topic opens in Help.

The following figure shows a tooltip with opened cascade:



Find what?

Whether or not a search is possible and which options are available depends on the open editor and the installed products.

Search and replace
Information on the search function

□ Wh
□ Ma
□ Fin
□ Fin
□ Us
□ Use regular expressions

○ Whole document
● From current position
○ Selection

● Down
○ Up

See also

Disabling the automatic display of tooltip cascades (Page 206)

6.2.2 Opening the Help system

Opening the Help system

You can open the Help system in the following ways:

1. In the "Help" menu, select the "Show help" command or press <F1> to display the corresponding help for the current context.

Or

1. Click on the link in a tooltip cascade to go directly to an additional point in the Help system.

6.2.3 Searching the Help system for keywords

Searching for keywords in the help text

To search the help topics for predefined keywords, follow these steps:

1. Click the "Show/hide table of contents" button in the help toolbar to display the table of contents.

   The table of contents is displayed and the "Index", "Search" and "Favorites" tabs are visible.
2. Open the "Index" tab.
3. Enter the search term in the input box or select the search term from the list of key words.
4. Click "Display".

6.2.4 Full-text searches

Full-text searches

To search the entire text for specific words, follow these steps:

1. Click the "Show/hide table of contents" button in the help toolbar to display the table of contents.

   The table of contents is displayed and the "Index", "Search" and "Favorites" tabs are visible.

2. Open the "Search" tab.
3. Type in your search term in the text box.
4. Refine your search if necessary using additional criteria:
   – Select "Search previous results" to start an additional search operation of your last search results only.
   – Select "Search for similar words" to find words that differ only slightly from your search term.
   – Select "Search titles only" to obtain only results that contain your search term in the title. The contents of the Help topics are ignored during the search.
5. Click on the arrow button to the right of the search field to use logic operations. The following logic operations are available:
   – Combine two or more search terms using the "AND" operator to find only Help topics that contain all the search terms in the text.
   – Combine two or more search terms using the "OR" operator to find only Help topics that contain one or more of the search terms in the text.
   – Combine two or more search terms using the "NEAR" operator to find only Help topics that contain terms in close proximity to each other (eight words).
   – Precede a word with the "NOT" operator to exclude Help topics from the search that contain this word.
6. Click on "List topics" to start the search.

   The results are now listed with title, position and ranking. The "Position" column shows the section in which the Help topic found is located. Sorting according to ranking is based on the position of the Help topics found in the table of contents and based on the number of hits in the Help topics.

6.2.5 Using favorites

Using favorites

You can save individual help topics as favorites. This saves you searching for the help topic a second time.

Saving favorites:

To save a page as a favorite, follow these steps:

1. Open the help topic or the chapter you want to save as a favorite.
2. Click the "Show/hide table of contents" button in the help toolbar to display the table of contents.

   The table of contents is displayed and the "Index", "Search" and "Favorites" tabs are visible.
3. Open the "Favorites" tab.
4. Click the "Add" button.

   The help topic or chapter is saved as a favorite and is available the next time you open the help system.

Calling up favorites:

To call up a page from the favorites, follow these steps:

1. Click the "Show/hide table of contents" button in the help toolbar to display the table of contents.

   The table of contents is displayed and the "Index", "Search" and "Favorites" tabs are visible.
2. Open the "Favorites" tab.
3. Select the topic you want to open from the list.
4. Click the "Display" button.

Deleting favorites

To delete an entry from the favorites, proceed as follows:

1. Click the "Show/hide table of contents" button in the help toolbar to display the table of contents.

   The table of contents is displayed and the "Index", "Search" and "Favorites" tabs are visible.
2. Open the "Favorites" tab.
3. Select the topic you want to remove from the list.
4. Click the "Remove" button.

6.2.6 Printing help topics

Printing information

You can either print all the contents of the Help system or individual topics only.

Procedure

To select the topics you would like to print, follow these steps:

1. Click the "Display printing dialog" button.

   The table of contents opens in a separate window.
2. Select the check boxes for the folders and help topics to be printed in the "Print help topics" dialog.
3. Click the "Print" button to print the selected information.

   The "Print" dialog opens.
4. Select the printer on which you want print the help topics.
5. Click "Properties" if you want to make additional printer settings.
6. Confirm your entries with "OK".

   The help topics are printed out on the selected printer.

6.2.7 Disabling the automatic display of tooltip cascades

You can suppress the automatic display of tooltip cascades. Manual display remains possible.

Procedure

To disable the automatic display of tooltip cascades, follow these steps:

1. In the "Options" menu, select the "Settings" command.
2. Select the "General" group in the area navigation.
3. Disable the "Open cascade automatically in tooltips" check box in the "General settings".

If you want to display a tooltip cascade manually, click on the arrow icon within the tooltip.

See also

General remarks on the information system (Page 200)

6.2.8 Safety Guidelines

Safety guidelines

This Help manual contains you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury will result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury may result if proper precautions are not taken.

⚠ CAUTION
with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.

CAUTION
without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.

NOTICE
indicates that an unintended result or situation can occur if the corresponding information is not taken into account.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The device/system may only be set up and used in conjunction with this documentation. Commissioning and operation of a device/system may only be performed by qualified personnel. Within the context of the safety notes in this documentation qualified persons are defined as persons who are authorized to commission, ground and label devices, systems and circuits in accordance with established safety practices and standards.

Prescribed Usage

Note the following:

⚠ WARNING
This device may only be used for the applications described in the catalog or the technical description and only in connection with devices or components from other manufacturers which have been approved or recommended by Siemens. Correct, reliable operation of the product requires proper transport, storage, positioning and assembly as well as careful operation and maintenance.

Trademarks

All names identified by ® are registered trademarks of the Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Editing projects 7

7.1 The basics of projects

Introduction

Projects are used to organize the storage of data and programs resulting from the creation of an automation solution. The data that makes up a project includes the following:

- Configuration data on the hardware structure and parameter assignment data for modules
- Project engineering data for communication over networks
- Project engineering data for the devices

Project hierarchy

Data is stored in a project in the form of objects. Within the project, the objects are arranged in a tree structure (project hierarchy).

The project hierarchy is based on the devices and stations along with the configuration data and programs belonging to them.

Common data of the project and online access, for example, are also displayed in the project tree.

See also

Creating a new project (Page 210)

Opening projects (Page 210)

Saving projects (Page 213)

Deleting projects (Page 214)

7.2 Creating and managing projects

7.2.1 Creating a new project

Procedure

To create a new project, follow these steps:

1. Select the "New" command in the "Project" menu.

   The "Create a new project" dialog opens.
2. Enter your project name and path or accept the proposed settings.
3. Click the "Create" button.

Result

The new project is created and displayed in the project tree.

See also

The basics of projects (Page 209)

Opening projects (Page 210)

Saving projects (Page 213)

Deleting projects (Page 214)

7.2.2 Opening projects

Procedure

To open an existing project, follow these steps:

1. Select the "Open" command in the "Project" menu.

   The "Open project" dialog opens and includes the list of most recently used projects.
2. Select a project from the list and click "Open".
3. If the project you require is not included in the list, click the "Browse" button. Navigate to the desired project folder, and open the project file. Projects of the TIA Portal V11 have the extension ".ap11", while older projects from the TIA Portal V10 have the extension "ap10".

Result

The project opens in the project view.

Notes on compatibility

When you open a project that was created with an older version of the TIA Portal, the project is converted automatically to the current file format and saved as a new project. The designation "_V11" is added to the file name of the new project.

If you want to open a project from a newer version, this is possible if the following conditions are met:

- The project was created with a different version of the TIA Portal V11, for example, a version with an installed service pack.
- The project does not contain any data that is incompatible with the current installation.

If the project to be opened contains data that was created with optional software, but the corresponding software product is not installed, the following cases can occur:

- Software components are missing, but none of them are essential:

  A dialog appears listing the missing software components. After the project is opened, its properties are displayed. You now have the opportunity to install the missing products. All the devices contained in the project are available even if you do not install the missing products. However, you can only work with the devices that are supported by the currently installed software.

  If devices are not supported because software is missing, they are marked with the following symbol in the project tree:

- At least one software package is required in order to open the project:

  A dialog appears listing the missing software components. The essential package(s) are marked. The project can only be opened if you install the missing components.

See also

The basics of projects (Page 209)

Creating a new project (Page 210)

Saving projects (Page 213)

Deleting projects (Page 214)

7.2.3 Displaying properties of the project

You can display the properties of a project. Properties include the following:

- Metadata for the project

  This includes the following information: creation time, author, file path, project size, copyright , project languages, etc. Many of the attributes can be changed.

- Project history

  The project history contains an overview with important events in the project life cycle. Here you can also call the log file for a migration.

- Support packages in the project

  You can display an overview of the additional software needed to work with all devices in the project.

- Software products in the project

  You can display an overview of all installed software products needed for the project.

Procedure

To display the project properties, follow these steps:

1. Select the open project in the project tree.
2. Select "Properties" in the shortcut menu of the project.

   The dialog with the properties of the project opens.
3. Select the project properties in the area navigation that you want to have displayed.

7.2.4 Saving projects

You can save the project at any time either under the same or a different name. You can even save a project when it still contains elements with errors.

Saving a project

To save a project, follow these steps:

1. Select the "Save" command in the "Project" menu.

   All changes to the project are saved under the current project name.

Project Save as

To save a project under another name, follow these steps:

1. Select the "Save as" command in the "Project" menu.

   The "Save current project as" dialog opens.
2. Select the project folder in the "Save in" box.
3. Enter the new project name in the "File name" box.
4. Confirm your entry with "Save".

   The project is saved under the new name and opened.

See also

The basics of projects (Page 209)

Creating a new project (Page 210)

Opening projects (Page 210)

Deleting projects (Page 214)

7.2.5 Closing projects

Procedure

To close a project, follow these steps:

1. Select the "Close" command in the "Project" menu.

   If you have made changes to the project since the last time you saved it, a message is displayed.
2. Decide whether or not you want to save the changes.

7.2.6 Deleting projects

NOTICE
When you delete a project, the entire project data is removed from the storage medium.

Requirements

The project you want to delete is not open.

Procedure

Follow the steps below to delete an existing project:

1. Select the "Delete project" command in the "Project" menu.

   The "Delete project" dialog opens and includes the list of most recently used projects.
2. Select a project from the list.

   If the project you require is not included in the list, click the "Browse" button. Navigate to the required project folder, and open the project file with the extension ".ap11".
3. Click the "Delete" button.
4. Click "Yes" to confirm. This starts the deletion of the project.

Result

The entire project folder is deleted from the file system.

See also

The basics of projects (Page 209)

Creating a new project (Page 210)

Opening projects (Page 210)

Saving projects (Page 213)

7.2.7 Working with multi-language projects

7.2.7.1 Project text basics

Texts in different languages in the project

When you enter texts while working on a project, you would normally do this in your own language. If you then pass on the project to someone else who does not know this language, this person will require a translation of the relevant texts to a language they know. This is why all texts can be translated. In this way, you can ensure that anyone who is subsequently confronted with the texts sees the texts in his/her language of choice.

Project language

Project languages are all languages in which a project will later be used. Based on the editing language, all the texts can be translated to the various project languages. You specify the languages that will be available in the project tree under "Languages & Resources > Project languages".

Editing language

Every project has an editing language. When you enter texts, these are always created in the editing language. You should therefore make sure that the editing language set is the language in which you enter the texts. This avoids problems if you translate the texts later.

The editing language does not depend on the language of the user interface. You could, for example, set English as the user interface language, but use Italian as the editing language. If you enter texts, these will be created in this case in the project language "Italian", although the user interface of the TIA Portal displays English.

You set the editing language in the project tree under "Languages & Resources > Project languages > Editing language".

User texts and system texts

For clarification purposes, a distinction is made between user texts and system texts:

- User texts are texts that the user created.
- System texts are texts that are created automatically according to the configuration in the project.

You manage the project texts in the project tree under "Languages & Resources > Project texts".

Examples of multilingual project texts

You can, for example, manage the following project texts in more than one language:

- Block titles and block comments
- Network titles and network comments
- Statement comments from STL programs
- Comments in tables
- Alarm texts
- Operator-relevant texts
- Text lists
- Labels of buttons
- Display names of recipes

Translating texts

There are three ways of translating texts.

- Translating texts directly

  You can enter the translations for the individual project languages directly in the "Project texts" table. You will find this in the project tree under "Languages & Resources > Project texts".

- Translating texts using reference texts

  You can change the editing language for shorter texts. All the text cells are filled again with the default values and can be filled in the current language. As orientation, you can display what you last entered in the box in the reference language. To do this, select the "Tasks" task card and open the "Languages & resources".

- Exporting texts and translating them externally

  With larger volumes of text, you can export the texts to an Office Open XML file and translate them with a normal table calculation program. You then import the translated texts again into the TIA Portal.

Note

Using Asian project languages

East Asian project languages are only displayed correctly in Windows XP, if the option "Install files for East Asian languages" is selected on the Languages tab of Regional and Language Options in the Control Panel of Windows XP Professional.

See also

Overview of the program settings (Page 162)

Changing the settings (Page 166)

Application examples for multilanguage projects (Page 222)

7.2.7.2 Select project languages

All the texts can be displayed within a project in the same language that you selected for your software user interface. This means that all project texts must exist in the corresponding language. You can select the available project languages yourself.

Requirement

- You are in the project view.
- A project is open.

Procedure

To select the project languages, follow these steps:

1. Click on the arrow symbol to the left of "Languages & Resources" in the project tree.

   The elements below this are displayed.
2. Double-click on "Project languages".

   In the work area, you will see a list of languages that you can select.
3. Select the required languages.

Result

All texts can be displayed in the activated languages if there is already a translation for these languages.

7.2.7.3 Setting the editing language

All the texts in the project are created in the editing language when they are entered. If you change the editing language, all future text input will be stored in the new editing language.

Requirement

- You are in the project view.
- A project is open.

Procedure

To change the editing language, follow these steps:

1. Click on the arrow symbol to the left of "Languages & Resources" in the project tree.

   The lower-level elements are displayed.
2. Double-click on "Project languages".

   The possible settings for the project languages are displayed in the work area.
3. Select the editing language in "General > Editing language".

7.2.7.4 Translating texts directly

If you use more than one language in your project, you can enter translations of individual project texts directly in the selected project languages. To allow this, there is a list for user texts and one for system texts with a column for every project language.

Requirement

- You are in the project view.
- A project is open.
- You have selected at least one further project language.

Procedure

To translate individual texts, follow these steps:

1. Click on the arrow symbol to the left of "Languages & Resources" in the project tree.

   The elements below this are displayed.
2. Double-click "Project texts".

   A list with the user texts in the project is displayed in the work area.
3. Click on "System texts" if you you want to edit the list of system texts rather than the user texts.
4. You can improve the clarity of the lists if you have a lot of texts.
   - To group identical texts and to translate them all at once, click the "Switch on/off grouping" button in the toolbar.
   - To hide texts that do not have a translation, click the "Filter for empty texts on/off" button in the toolbar.
   - To further limit the displayed project texts to certain devices, select the devices for which you want to display project texts in the drop-down list.
5. Enter the translation of the project texts in the relevant column.

Note

If there is no translation for a text in a particular language, the English text is displayed.

7.2.7.5 Translating texts using reference texts

Introduction

After changing the editing language, all texts are shown in input boxes in the new editing language. If there is not yet a translation available for the newly set language, the input boxes are empty or filled with default values.

If you enter text in an input box, this is saved in the current editing language. Following this, the texts exist in two project languages for this input field, in the previous editing language and in the current editing language. This makes it possible to create texts in several project languages.

You can display existing translations for an input box in other project languages. These serve as a comparison for text input in the current editing language and they are known as the reference language.

Note

The display of reference texts depends on the installed products and is not supported by every editor.

Requirement

There is at least one translation into a different project language for an input field.

Procedure

To display the translation of an input cell in a reference language, follow these steps:

1. In the "Tasks" task card, select the "Languages & Resources" pane.
2. Select a reference language from the "Reference language" drop-down list.

Result

The reference language is preset. If you click in a text box, translations that already exist in other project languages are shown in the "Tasks > Languages & Resources" task card.

See also

Application examples for multilanguage projects (Page 222)

7.2.7.6 Exporting and importing project texts

You can export project texts for translation and then reimport them. The texts are exported to an Office Open XML file with the extension ".xlsx". This can be edited in Microsoft Excel or a number of other spreadsheet programs.

The following export options are available:

- Exporting individual project texts
- Exporting all user texts or system texts at once

  In this case, the export can be additionally limited by categories.

Note

Row limit in Microsoft Excel

Note that Microsoft Excel 2003 supports a maximum of 65536 rows. Microsoft Excel 2007 supports up to 1048576 rows.

Exporting individual project texts

To export individual project texts, follow these steps:

1. Open the "Languages & Resources" folder in the project tree.

   The lower-level elements are displayed.
2. Double-click "Project texts".

   The project texts editor opens.
3. Choose the "User texts" or "System texts" tab in the editor, depending on which texts you want to export.
4. Select the project texts you want to export.
5. Click the "Export project texts" icon in the toolbar of the editor.

   The "Export" dialog box opens.
6. Choose the language you want to translate from in the "Source language" drop-down list.
7. Choose the language you want to translate to in the "Target language" drop-down list. The drop-down list contains the project languages you specified previously. If the required language is missing, you must first specify it in the project languages editor.
8. Specify a file path and a file name for the export file in the "Select file for export" input box.
9. Click "Export".

Exporting all system or user texts

To export all project texts, follow these steps:

1. Select the "Export project texts" command in the "Tools" menu.

The "Export" dialog box opens.

2. Choose the language you want to translate from in the "Source language" drop-down list.
3. Choose the language you want to translate to in the "Target language" drop-down list. The drop-down list contains the project languages you specified previously. If the required language is missing, you must first specify it in the project languages editor.
4. In "Select content", select the check box "User texts" to export user texts. To export system texts, select "System texts". To export both user texts and system texts, select both check boxes.
5. In "Select content", select the required text categories for the user texts or the system texts.
6. In the "Export file" input field, specify a file name for the export file.
7. In the "Path" input field, select a path in the data system to which the export file is to be saved.
8. Click "Export".

Importing project texts

To import a file containing project texts, follow these steps:

1. Select the "Import project texts" command in the "Tools" menu.

   The "Import" dialog box opens.
2. Select the path and the file name of the import file from the "Select file for import" field.
3. Select the "Import base language" check box if you have made changes to the base language in the export file and you want to overwrite the entries in the project with the changes.
4. Click "Import".

See also

Application examples for multilanguage projects (Page 222)

7.2.7.7 Application examples for multilanguage projects

Introduction

Let us assume you are working in a team with colleagues some of whom speak English, some French and some German. You have created a project with the TIA Portal and have already created a functioning configuration.

To allow your other colleagues to be able to keep track of the project, you would like all devices being used to have comments in English and German. First, you would like to enter the comments in German. Following this, to save time and costs, you want to have the texts translated into English in a spreadsheet program by an external translation office.

In addition to this, you also want a single comment for a particular device in French so that your French-speaking colleague can continue working on this device.

The section below describes an example of how you can achieve this with the tools of the TIA Portal.

Translating the project into English

To enter the comments in German and to have them translated into English later, follow these steps:

1. Set the editing language to "German" and fill all the comment boxes with the relevant texts in German.

   On the device selected from the French-speaking colleague, enter, for example "Unser neues Gerät" in German first.

   All the comments are now stored in German.
2. Export all user texts to an Office Open XML file with the extension ".xlsx".
3. Have the user texts contained in the file translated into English in a spreadsheet program such as Microsoft Excel.
4. Import the file into the TIA Portal after it has been translated.

   All texts are now available in German and English.

Translating a single comment field to French

To translate an individual comment field to French, follow these steps:

1. Open the comment box for the device on which the French-speaking colleague will be working.
2. Open the "Languages & Resources" pane in the "Tasks" task card.
3. Set "French" as the editing language in the "Languages & Resources" pane. As the reference language, set, for example, "English".

   Since no translation has yet been installed in French, the comment box is empty. In the "Languages & Resources" pane, the English translation "Our new device" is displayed as a reference.
4. Orientating yourself on the English reference text enter "Notre nouvel appareil" in the comment box.

   The comment for this device is now available in the languages German, English and French.

See also

Project text basics (Page 215)

Exporting and importing project texts (Page 220)

Translating texts using reference texts (Page 219)

7.3 Editing project data

7.3.1 Compiling and loading project data

7.3.1.1 Compiling project data

General information on compiling project data

Compiling project data

During compilation, project data is converted so that it can be read by the device. Hardware configuration data and program data can be compiled separately or together. You can compile the project data for one or more target systems at the same time.

The following project data must be compiled prior to loading:

- Hardware project data, for example, configuration data of the devices or networks and connections
- Software project data, for example, program blocks or process screens

Scope of the compilation

When you compile project data, you have the following options depending on the device involved:

- All
- Hardware configuration
- Software
- Software (rebuild all blocks)

See also

Compiling project data (Page 224)

Compiling project data

The following section describes the general procedure for compiling project data in the project tree. You will find details of how certain objects are compiled and any special points to note in the online help of the product.

Procedure

To compile project data, follow these steps:

1. In the project tree, right-click on the device for which you want to compile the project data.

2. Select the option you require in "Compile" submenu of the shortcut menu.

Note

Note that the options available to you depend on the selected device.

The project data is compiled. You can check whether or not the compilation was successful in the Inspector window with "Info > Compile".

Canceling compilation operation

You can cancel compilation at any time. Proceed as follows:

1. Click the "Cancel" button in the status bar next to the progress bar. If the TIA Portal is currently processing multiple asynchronous processes, you can use the Up or Down arrow to change to the progress display of the compilation operation you want to cancel.
2. Click "Yes" to confirm.

See also

General information on compiling project data (Page 224)

7.3.1.2 Downloading project data

General information on loading

Introduction

In order to set up your automation system, you must download the project data you generated offline to the connected devices. This project data is generated, for example when configuring hardware, networks, and connections or when programming the user program or when creating recipes. The first time you download, the entire project data is downloaded. During later loading operations, only changes are downloaded.

You can download the project data to the following destinations:

- Devices
- Accessible devices
- Memory cards

Depending on the object you want to download, you have the following options:

- All

  Both hardware configuration as well as software are downloaded to the destination.
- Hardware configuration

  Only the hardware configuration is downloaded to the destination.
- Software

  Only the objects that differ online and offline are downloaded to the destination.
- Software (all blocks)

  All blocks are downloaded to the destination.

You can also upload project data already contained in a device back to your project. You have the following options:

- Uploading a complete device

  All relevant data of the device is uploaded to the project.
- Uploading blocks and parameters

  Only the blocks and parameters from the device are uploaded to the project.

See also

Downloading project data to a device (Page 227)

Loading project data to an accessible device (Page 228)

Downloading project data to a memory card (Page 229)

Uploading project data from a device (Page 230)

Downloading project data to a device

The following section describes the general procedure for downloading project data to a device. You will find details of how certain objects are downloaded and any special points to note in the online help of the product.

Requirement

- The project data is consistent.
- Each device to which you want to download is accessible via an online access.

Procedure

To download the project data to the selected devices, follow these steps:

1. Select one or more devices systems in the project tree.
2. Right-click on a selected element.

   The shortcut menu opens.
3. Select the option you require in the shortcut menu of the "Download to device" submenu.

   Note

   Note that the options available to you depend on the selected device.

   When necessary, the project data is compiled.

   – If you had previously established an online connection, the "Load preview" dialog opens. This dialog displays messages and proposes actions necessary for downloading.

   – If you had not previously established an online connection, the "Extended download to device" dialog opens, and you must first select the interfaces via which you want to establish the online connection to the device.

     See also: Establishing and canceling an online connection (Page 3074)
4. Check the messages in the "Load preview" dialog, and select the actions in the "Action" column, if necessary.

   NOTICE

   Performing the proposed actions while the plant is in operation can cause serious bodily injury and property damage in the event of malfunctions or program errors.

   Make sure that no dangerous situations can arise before you start the actions!

As soon as loading becomes possible, the "Load" button is enabled.

5. Click the "Load" button.
6. The loading operation is performed. The "Load results" dialog then opens. In this dialog, you can check whether or not the loading operation was successful and take any further action that may be necessary.
7. Click the "Finish" button.

Result

The selected project data was downloaded to the devices.

See also

General information on loading (Page 226)

Loading project data to an accessible device (Page 228)

Downloading project data to a memory card (Page 229)

Uploading project data from a device (Page 230)

Loading project data to an accessible device

The following section describes the general procedure for downloading project data to an accessible device in the project tree. You will find details of how certain objects are downloaded and any special points to note in the online help of the product.

Requirement

The accessible devices are displayed.

See also: Displaying accessible devices (Page 3071)

Procedure

To load project data to an accessible device, follow these steps:

1. In the project tree, drag the folder containing your device to the accessible device.

   The "Load preview" dialog opens. This dialog displays messages and proposes actions necessary for downloading.

2. Check the messages, and select the actions in the "Action" column, if necessary.

NOTICE
Performing the proposed actions while the plant is in operation can cause serious bodily injury and property damage in the event of malfunctions or program errors. Make sure that no dangerous situations can arise before you start the actions!

3. As soon as loading becomes possible, the "Load" button is enabled.

4. Click the "Load" button.

   The loading operation is performed. The "Load results" dialog then opens. In this dialog, you can check whether or not the loading operation was successful and take any further action that may be necessary.

5. Click the "Finish" button.

See also

General information on loading (Page 226)

Downloading project data to a device (Page 227)

Downloading project data to a memory card (Page 229)

Uploading project data from a device (Page 230)

Downloading project data to a memory card

Requirement

The memory card is displayed.

See also: Accessing memory cards (Page 270)

Procedure

To download project data to a memory card, follow these steps:

1. Use a drag-and-drop operation in the project tree to take the project data you want to download and move it to the memory card.

   The "Load preview" dialog opens. This dialog displays messages and proposes actions necessary for downloading.
2. Check the messages, and select the actions in the "Action" column, if necessary.

   As soon as loading becomes possible, the "Load" button is enabled.
3. Click the "Load" button.

   The loading operation is performed. The "Load results" dialog then opens. In this dialog, you can check whether or not the loading operation was successful and take any further action that may be necessary.
4. Click the "Finish" button.

See also

General information on loading (Page 226)

Downloading project data to a device (Page 227)

Loading project data to an accessible device (Page 228)

Uploading project data from a device (Page 230)

Uploading project data from a device

The following section describes the general procedure for uploading project data from a device. Which project data you can upload from a device depends on the products installed.

You have the following basic options for uploading project data from a device to your project:

- Uploading a device to a programming device or PC

  You can use this option to start with an empty project and upload existing project data directly from a device.
- Uploading from device

  Only certain project data are uploaded from the device to the project. You will find the project data that can be downloaded in the online help of the product.

Requirement

- A project is open.
- The hardware configuration and software to be downloaded have to be compatible with the TIA Portal. If the data on the device was created with a previous program version or with a different configuration software, please make sure they are compatible.

Uploading a device to a programming device or PC

To upload the complete device to your project, follow these steps:

1. Select the project name in the project tree.

   The "Upload device to PG/PC" command in the "Online" menu is then enabled.
2. In the "Online" menu, select the "Upload device to PG/PC" command.

   The "Upload device to PG/PC" dialog opens.
3. Select the type of interface you want to use for the load operation in the "Type of the PG/PC interface" drop-down list.
4. Select the interface to be used from the "PG/PC interface" drop-down list.
5. Click the "Configure interface" button to the right of the "PG/PC interface" drop-down list to adapt the settings for the selected interface.

   See also: Establishing and canceling an online connection (Page 3074)
6. In the accessible devices table, select the device from which you want to upload project data.
7. Click on "Load".

   Depending on the selected device, a dialog appears in which you have to enter additional information, such as the position of the module rack.

   The project data of the device is uploaded to the project. You can edit it offline and then download it to the device again.

Or:

1. Open the accessible devices in the project tree.

   See also: Displaying accessible devices (Page 3071)
2. Use a drag-and-drop operation in the project tree to move the accessible devices to the project.

   Depending on the selected device, a dialog appears in which you have to enter additional information, such as the position of the module rack.

   The project data of the device is uploaded to the project. You can edit it offline and then download it to the device again.

Uploading from device

To upload only certain project data from a device to your project, follow these steps:

1. Establish an online connection to the device from which you want to download the project data.

   See also: Establishing and canceling an online connection (Page 3074)
2. Select an element in the project tree that allows uploading of project data.

   As a result, the "Upload from device" command in the "Online" menu becomes enabled.
3. Select the "Upload from device" command in the "Online" menu.

   The "Upload preview" dialog box opens.
4. Check the messages in the "Upload preview" dialog, and select the necessary actions in the "Action" column.

   As soon as uploading becomes possible, the "Upload from device" button is enabled.
5. Click the "Upload from device" button.

   The loading operation is performed.

See also

General information on loading (Page 226)

Downloading project data to a device (Page 227)

Loading project data to an accessible device (Page 228)

Downloading project data to a memory card (Page 229)

7.3.2 Comparing project data

7.3.2.1 Basics of project data comparison

Function

You have the option of comparing project data of the same type in order to determine possible discrepancies. For example, you can compare a hardware configuration with another hardware configuration.

A simple online-online comparison is already performed when you establish an online connection. During this process, comparable objects in the project tree are marked with icons that represent the result of the comparison. Beyond this marking, you can use the comparison editor to compare project data and have detailed information displayed. You can also select actions for non-identical objects in the comparison editor.

In addition to the online-offline comparison, there is also an offline-offline comparison available. This allows you to compare project data of two devices within one project or from different projects.

Which project data you can compare depends on the products installed.

Comparison icons

The result of the comparison is indicated by means of icons. The following table shows the relevant icons and their meaning:

Icon	Meaning
❗	Folder contains objects whose online and offline versions differ
❓	Comparison result is not known
🟢	Online and offline versions of the object are identical
◐	Online and offline versions of the object are different
◖	Object only exists offline
◗	Object only exists online

7.3.3 Protecting project data

7.3.3.1 Protection concept for project data

Introduction

You can protect your project data from unauthorized access. These include, for example:

- Access protection for devices
- Copy and display protection of objects
- Restrictions for printouts of know-how-protected objects

Note that every protection mechanism is not available for all objects. How to protect specific objects is described in the online help of the product.

Revoking access rights for devices

If you want to execute a function that is password-protected by means of the device protection level, you are prompted to enter a password. When the password is entered correctly, you can execute the required function. You continue to have access rights on the device until you close the TIA Portal.

If you want to reactivate password protection while the TIA Portal is open, you can explicitly revoke the access rights for a device. As a result, certain functions for the protected device cannot be executed until the correct password is entered again. You specify the functions for which a password must be entered when you assign the device protection level.

See also

Printing project data (Page 254)

7.3.3.2 Revoking access rights for devices

Requirement

- A protection level has been set for the device.
- A protected function for the device has been enabled by entering the password.

Procedure

To revoke the access rights for the device, follow these steps:

1. Select the device for which you want to revoke access rights in the project tree.
2. Select the "Delete access rights" command in the "Online" menu.

Result

The access rights are revoked, and starting from now the user will be prompted to enter the password again to execute a password-protected function on the device. The function can only be executed if the correct password is entered.

If the device has an online connection, it will be disconnected.

See also

Protection concept for project data (Page 234)

7.3.4 Printing project contents

7.3.4.1 Documentation settings

Introduction

Once a project is created, the contents can be printed in an easy-to-read format. You may print the entire project or individual objects within the project. A well-structured printout is helpful when editing the project or performing service work. The printout can also be used for your customer presentations or as full system documentation.

You can prepare the project in the form of standardized circuit manuals and print it in a uniform layout. You can limit the scope of the printout. You have the option to print to the entire project, individual objects along with their properties, or a compact overview of the project. In addition, you can print the contents of an open editor.

Improving the printout with frames and cover pages.

You can design the appearance of the printed pages according to your own requirements, for example, with your company logo or company layout. You can create your own printout templates with frames that will surround the project data in the printout. You create the frames in an external layout program or an image editing program and save the file in PDF or EMF format. The file can then be imported as a background image. You can insert placeholders for data from previously entered document information on the background. These will be filled automatically with the appropriate metadata during printing.

If you want to avoid designing your own template, there are ready-made frames and covers pages available. These include templates complying with the ISO standard for technical documentation.

If you already have cover pages or frames from earlier versions of the TIA Portal or you have templates and cover pages in EMF or PDF file format, you can import these.

Modular structure of a printout

An printout generally consists of the following components:

- Cover page (only when printing from the project tree)
- Table of contents (only when printing from the project tree)
- Name and path of an object within the project tree
- Object data

Printout of the cover page or the table of contents can be deactivated in the "Print" dialog.

See also

Creating a frame (Page 242)

Creating a cover page (Page 242)

Editing cover pages and frames (Page 244)

Entering document information (Page 240)

7.3.4.2 Printout of project contents

Availability of print function

The following contents can be printed:

- An entire project in the project tree
- One or more project-related objects in the project tree
- Contents of an editor
- Tables
- Libraries
- Diagnostics view of the Inspector window

It is not possible to print in the following areas:

- Portal view
- Detailed view
- All tabs of the Inspector window, except the diagnostics view
- All task cards, except the libraries
- Most of the dialogs

Scope of printout

To be able to print, at least one printable element has to be selected.

If a selected object is printed, all subordinate objects are also printed. For example, if a device is selected in the project tree, all of its data is also printed.

When table contents are printed, all lines in the table in which a cell is selected are printed. In order to print one or more table columns, the desired columns must be selected. If no individual cells or columns are selected, the entire table is printed.

Limitations when printing

In general, it is possible to print all objects that can be visualized on the user interface. Conversely, this means that you cannot print objects that you do not have access to. If a printout fails, possible reasons may include the following:

- A valid license does not exist for displaying an object.
- There is no device description for an object.
- A software component needed to display an object is not installed.

See also

Printing project data (Page 254)

7.3.4.3 Changing the print settings

Changing the print settings

You can specify general print settings that are retained even after the TIA Portal is closed and re-opened. Some settings are dependent on the products installed. The following settings are possible in every case:

Always print table data as pairs of values

If this option is selected, tables are not printed in tabular format but rather as a pairs of key and value.

Example:

Object name	Property 1	Property 2
Object A	Value A1	Value A2
Object B	Value B1	Value B2

In this case, the printout has the following appearance:

Object A

Property 1: Value A1

Property 2: Value A2

Object B

Property 1: Value B1

Property 2: Value B2

Printing mask editors

- Always print data in tables

  All parameters of technology objects are printed in tabular format.

- Print mask graphics if possible

  If the utilized editor supports this function, the contents of the editor are not printed as a table but rather as a complete graphic as it appears on the screen.

Procedure

To change the print settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.

   The "Settings" window is displayed in the work area.
2. Select the "General" group.
3. Select the desired default settings in the "Print settings" area.

   The changes are applied immediately and are retained for all projects, even after the TIA Portal is closed.

See also

Overview of the print settings (Page 165)

7.3.4.4 Specifying the print layout

Specifying the print layout

If you do not want to rely on ready-made print templates, you can specify your own cover page or your own layout for the individual pages. Your designs are saved together with the respective project.

Your designs for the cover page and your templates for the page layout can be found in the project tree under the "Documentation information" group. You will also find metadata on the project there under the entry "Document information". For subsequent print operations, you can customize the appearance of the printout in the "Print" dialog using the saved cover pages and page layout templates and the available metadata.

Designing the cover page

The cover page can be customized. You can insert a background graphic and provide placeholders for text on the page. The placeholders are automatically filled with data from a documentation information during printing.

Cover pages are located in the project tree under the "Documentation information > Cover pages" group.

Designing the content page

The regular pages of a printout can contain the following elements:

- Frame with static content, such as a company logo
- Place holders for text, such as the name of the project, the page number, and the time the printout was started

  Several different values for the individual placeholders can be specified in the document information. Other values, such as the project name, are preassigned and are inserted automatically during printing.
- Footnote

  The footnote is always output below the content area.
- Content area

  You can specify an area where the printed content is to be embedded.

The design of the content pages is saved in Frames. The individual frames are located in the project tree under the "Documentation information > Frames" group.

7.3.4.5 Entering document information

You can enter metadata in the document information for every project. In addition, a print frame and a cover page are specified in the document information. You can create different information, if required, to enable you to quickly switch between different document information containing different information, frames, cover pages, page sizes, and page orientations when printing. For example, this is useful if you want to generate printouts in different languages and different document information is provided for each language.

In the documentation editor, you can specify placeholders on the cover page or in the frame of the regular pages. These placeholders can be automatically replaced with metadata from the documentation information during printing.

The various document information are therefore part of the printing function and specify the print layout and print content.

Procedure

To add metadata, follow these steps:

1. To create new document information, double-click "Add new document information" under "Documentation information > Document information" in the project tree.

   The new document information is created and opened immediately.
2. Enter a name for the set in the "Name" field.
3. Fill in the individual fields with the metadata for the project.

7.3.4.6 Managing cover pages and frames

Using cover pages and frames

Uses for cover pages

You can give your plant documentation printouts a professional appearance by adding a cover page. You can design your own cover page or use ready-made cover pages. Ready-made cover pages can be adapted and stored again as a template.

Cover pages can be saved in global libraries where they are available for use across projects.

Cover pages are designed for use as a right printed page only.

Uses of frames

You can embed the regular pages of your plant documentation inside a consistently uniform page frame. The frame can contain placeholders for project metadata, which is stored in the document information. It can also contain graphic elements that you design yourself.

You can create your own frames or rely on ready-made page frames. You can adapt a ready-made page frame and then store it again as a new frame.

Like cover pages, frames can be saved in global libraries where they are available for use across projects.

Frames are designed for use on right printed pages only.

Cover pages and templates in the project tree

Cover pages and frames associated with the project are stored in the project tree under the entry "Documentation information". There are separate folders here for frames and cover pages.

The following actions are available in the project tree for cover pages and frames.

- Creating your own subfolders
- Copying and pasting
- Inserting cover pages and frames from the "Documentation templates" system library
- Copying cover pages and templates to a global library

Cover pages and templates in libraries

The "Documentation templates" system library contains a few cover pages and templates that are available in every project. The cover pages and templates can be moved from there to the project tree using a drag-and-drop operation. You can then adapt the cover pages and templates in the project tree according to the requirements of your project.

Cover pages and templates can be moved from the project tree to a global library. Afterwards, these are available in every project.

See also

Library basics (Page 272)

"Libraries" task card (Page 273)

Global library basics (Page 285)

Using ready-made frames and cover pages (Page 243)

Creating a frame

You can create any number of frames for each project. The frames are stored in the project tree below the "Documentation information > Frames" group. You can assign a frame to all document information. When you select document information for printing, its associated frame is used.

Procedure

To create a new frame, follow these steps:

1. Double-click the entry "Add new frame" below the "Documentation information > Frames" group in the project tree.

   The "Add new frame" dialog opens.
2. Enter a name for the frame in the "Name" field.
3. Choose the paper size from the "Paper type" drop-down list.
4. Choose whether the page is to be created in portrait or landscape format in the "Orientation" drop-down list.

Click the "Add" button.

Result

A new frame is created. The frame is then opened automatically in the documentation editor where it can be edited.

See also

Editing cover pages and frames (Page 244)

Creating a cover page (Page 242)

Creating a cover page

You can create any number of cover pages for the printout for each project. The cover pages are stored in the project tree below the the "Documentation information > Cover pages" group. You can assign a cover page to all document information. When you select specific document information for printing, its associated cover page is used.

Procedure

To create a new cover page, follow these steps:

1. Double-click the entry "Add new cover page" below the "Documentation information > Cover pages" group in the project tree.

   The "Add new cover page" dialog box opens.
2. Enter a name for the cover page in the "Name" field.
3. Choose the paper size from the "Paper type" drop-down list.
4. Choose whether the page is to be created in portrait or landscape format in the "Orientation" drop-down list.

Click the "Add" button.

Result

A new cover page is created. The cover page is then opened automatically in the documentation editor where it can be edited.

See also

Editing cover pages and frames (Page 244)

Creating a frame (Page 242)

Using ready-made frames and cover pages

The TIA Portal comes with some ready-made frames and cover pages. These can change according to your wishes.

Procedure

To create and edit the ready-made frames and cover pages, follow these steps:

1. Open the "Global libraries" pane in the "Libraries" task card.
2. In the "Templates" folder, open the "Cover Pages" or "Frames" folder.
3. Drag a cover page or a frame from one of the folders into the project tree and drop it into one of the following folders:
   – For frames: "Document information > Frames"
   – For cover pages: "Document information > Cover pages".

   The ready-made frame or cover page can now be used in the project.
4. Double-click on the new entry in the project tree click to edit the frame or the cover page.

See also

Using cover pages and frames (Page 241)

Editing cover pages and frames (Page 244)

7.3.4.7 Designing cover pages and frames

Editing cover pages and frames

The documentation editor is a graphical editor which allows you to design frames and cover pages for your plant documentation. You can place images or text elements on the frame and the cover pages in the document editor. The text elements are either static or they are automatically filled during printing with the data from the document information that you have selected in the print dialog.

Procedure

To edit a cover page or a frame in the documentation editor, follow these steps:

1. In the project tree, double-click on the entry for an existing cover page or frame under the "Documentation information > Frames " or "Documentation information > Cover pages" group.

   The documentation editor opens.
2. Design the cover page or frame as desired.
3. Close the documentation editor.

   The changes to the cover page or frame are applied automatically.

See also

Creating a cover page (Page 242)

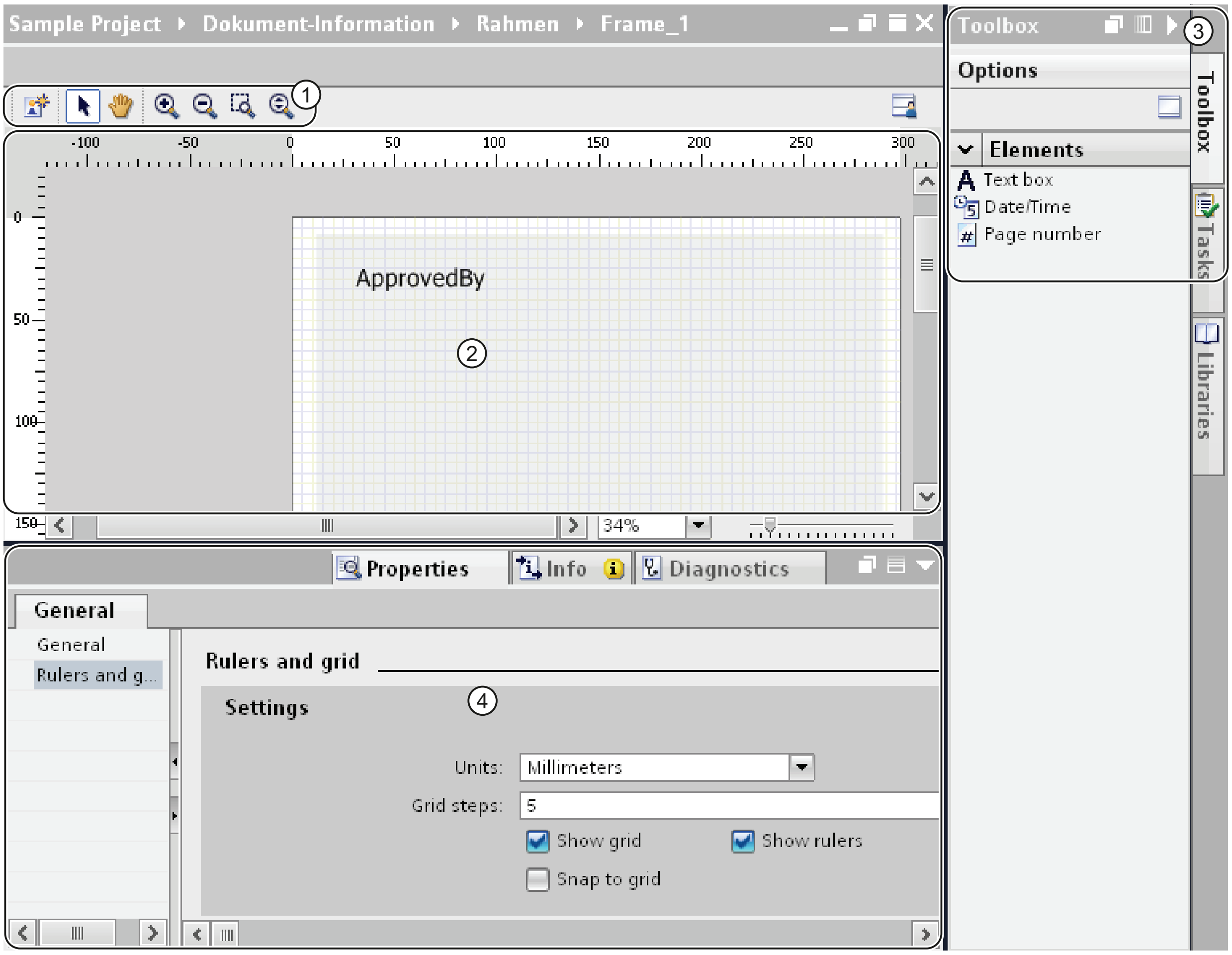
Creating a frame (Page 242)

General operation of the documentation editor (Page 245)

General operation of the documentation editor

Components of the documentation editor

The following figure provides an overview of the components of the documentation editor:



Units: Millimeters

Grid steps: 5

☑ Show grid ☑ Show rulers

☐ Snap to grid

① Toolbar

The toolbar provides the following tools (from left to right):

- Insert background image

  Inserts a background image into the template or cover page.
- Arrow tool

  Enables object selection.
- Navigation tool

  Allows shifting of the partial page.
- Zoom-in button

  Magnifies the page display incrementally.
- Zoom-out button

  Reduces the page display incrementally.
- Selecting a zoom factor

  Adapts the page size to the area selected with the lasso zoom tool.
- Dynamic zoom

  Adapts the page width to the work area.

② Work area

You can design the cover page or frame in the work area.

③ "Toolbox" task card

The "Toolbox" task card contains various types of placeholders that you can use on the cover sheet or frame. The placeholders can be placed in the work place using a drag-and-drop operation.

④ Properties in the Inspector window

You can display and modify the properties of the currently selected object in the "Properties" tab of the Inspector window. For example, you can modify the properties of the page, format text, specify the position of objects on the page, etc.

Operation in the documentation editor

The following basic functions are available in the documentation editor:

- Drag-and-drop functionality

  The documentation editor is a graphic editor, which means you can place objects anywhere with the mouse. An image of the page is displayed in the work area. This image corresponds to the ultimate print layout.

  If you want to select objects on the page in order to move them or modify their properties, the arrow tool must be activated in the toolbar.

- Zoom function

  You can use the zoom function to change the size of the page display. You have two options for changing the page size:

  – Via the buttons in the toolbar

    Select the "Zoom in" or "Zoom out" magnifying glass button in the toolbar of the documentation editor. Then click on the page in order to magnify (zoom in) or reduce (zoom out) the page incrementally.

    To zoom in on a particular area, select the "Select zoom factor" tool and use the mouse to drag an outline around the area you want to focus on.

    To continuously zoom in or zoom out of the work area, use the "Dynamic zoom" tool. To magnify the page display, click anywhere on the work area, and then hold down the mouse button while dragging the mouse toward the top of the page. To reduce the page display, drag the mouse toward the bottom of the page.

  – Via the zoom bar

    You can also use the zoom bar (located in the bottom right corner of the work area) to change the display size. Choose a percentage value from the drop-down list or enter a percentage value. Alternatively, you control the display size using the slider.

- Navigation over the page

  In addition to scrolling, you the option of changing the partial page with the navigation tool. To change the partial page with the navigation tool, select the Hand button in the toolbar. Then, click anywhere on the page and hold the mouse button down while moving the page to the desired position.

Using and adapting the positioning aids

You have various aids at your disposal to help you position elements on the page:

- Rulers

  Rulers are affixed to the page margins in the work area.
- Page grid

  A grid is placed underneath the page in the work area.

You can display, hide or adapt the positioning aids in the Inspector window under "Properties > Rulers and grid". You can make the following settings:

- Units:

  Specify the unit of measurement for the grid and the rulers.
- Grid steps:

  Specify the width of the grid.
- Show grid:

  Specify whether the grid is to be displayed or hidden.
- Snap to grid:

  Specify whether objects are to be aligned automatically to the grid. If the option is selected, the grid lines function like a "magnet".
- Show rulers:

  Specify whether the rulers are to be displayed.

See also

Editing cover pages and frames (Page 244)

Inserting a background image (Page 248)

Specifying the print area (Page 249)

Inserting placeholders for metadata (Page 250)

Inserting a background image

You can specify a background image for a cover page or frame. You can design the background image in another program and then insert it.

The background image is inserted in its original size. To keep it from extending beyond the margins, make sure that the background image is already correctly sized when you save it.

Requirement

- The background image must be in EMF or PDF format.
- A cover page is open in the documentation editor.

Procedure

To insert a background image, follow these steps:

1. Click the "Add background image" button.

   The "Open" dialog box opens.
2. Select the background image in the file system.
3. Click the "Open" button.

   The background image is inserted in its original size at the top left page margin.

See also

Creating a cover page (Page 242)

General operation of the documentation editor (Page 245)

Specifying the print area

An area within the frame is provided for the actual printed contents. The project data is then always inserted inside this defined and uniformly consistent area within the frame. You can adjust the size of the print area.

Requirement

A frame is open in the documentation editor.

Procedure

To define an area for the printed contents, follow these steps:

1. Click on the slightly darker area within the page display in the documentation editor to select the area for the print content.

   This opens the properties of the area to be printed in the Inspector window.
2. Enter the position of the print area on the X and Y axes in the Inspector window.
3. Specify the width and height of the print area in cm in the Inspector window.

Alternatively, you can change the width and position of the print area in the graphic display of the page. To do so, use the mouse to drag the margins of the print area until the desired size and position are achieved.

See also

Creating a frame (Page 242)

General operation of the documentation editor (Page 245)

Inserting placeholders for metadata

You can provide placeholders on the cover page and in a frame. The placeholders are automatically filled with metadata from documentation information during printing, if they are placeholders for text. Alternatively, you can add non-modifiable data, such as free text or an image.

All elements are arranged in numbered Z-Orders. If objects overlap, you can determine in which sequence these are to be arranged.

Types of placeholders

The following types of placeholders are available to you:

- Text field

  The text field stands as a placeholder for a text element from a document information. In the properties of the text field, you set which text from a document information should be automatically inserted during printing.
- Field for date and time

  A date and time is inserted instead of the placeholder when printing. This can be the date of creation or the point in time when the last change was made to the project. In the properties of the Inspector window, you specify which date or time is printed.
- Page number

  The correct page number is automatically applied when printing.
- Free text

  You can enter freely selectable text in the properties of the text field. The text is static and is not influenced by the document information selected at the time of printing.
- Image

  Select the image file in the properties of the placeholder in the Inspector window. Images in the formats BMP, JPEG, PNG, EMF or GIF are possible.

Requirement

An cover page or frame is open in the documentation editor.

Procedure

To insert placeholders for metadata on the cover sheet or in a frame, follow these steps:

1. Drag a field from the "Toolbox > Elements" task card to the work area of the documentation editor.

   The placeholder is inserted. The placeholder properties are shown in the Inspector window and can be edited there.
2. Select the metadata to be inserted during printing from the "Text" drop-down list in the Inspector window under "Properties > General > Text box". Alternatively, you have the option of entering free text or selecting an image depending on the type of placeholder.
3. In the Inspector window under "Properties > General > Position and size", specify the position of the placeholder on the X and Y axis and enter the width and height of the text box in cm. You specify the sequence of the objects in the "Z-Order" field, if these overlap. The smaller the value, the further down an object is located.
4. In the Inspector window, go to "Properties > View" and select the font formatting and the orientation of the text as well as the alignment of the text. You cannot make this setting for images.

See also

General operation of the documentation editor (Page 245)

7.3.4.8 Displaying print preview

Creating a print preview

Creating a print preview

You can create a preview of the printout. Document information can be chosen for this, in the same way as as for the actual printout. In this way, you preview the selected frame and, if applicable, the cover sheet. The settings are retained for later printing.

Procedure

To create a print preview and to set the scope of the later printout, follow these steps:

1. Select the "Print preview" command in the "Project" menu.

   The "Print preview" dialog opens.

2. Select the frame layout you want to use for the printout.

   – In the "Document information" drop-down list, select the documentation information you want to use later for the printout.

   – Select the "Print cover page" check box to print the cover page, which is specified in the selected document information.

   – Select the "Print table of contents" check box to add a table of contents to the printout.

   The check boxes for printing the cover page and the table of contents can only be selected if you have started the printout in the project tree.

3. Under "Print objects/area", select what is to be printed. The selection is only possible if you have started the printout from an editor that supports this function.

   – Choose "All" to print out the entire content of the editor.

   – Choose "Selection" to print only the objects currently selected in the editor.

4. Select the print scope under "Properties".

   – Choose "All" to print all configuration data of the selected objects.

   – Choose "Visible" to print the information of an editor that is currently visible on the screen. This option can only be chosen if you have started the printout from an editor that supports this function.

   – Choose "Compact" to print out an abbreviated version of the project data.

5. Click "Preview" to generate the preview.

   A print preview is created in the work area.

   Note

   Wait time for extensive documents

   It can take several minutes to generate the print preview in the case of very extensive projects. You can continue working normally in the meantime on systems with adequate resources. The progress of the print preview is shown in the status bar.

See also

Operation within the print preview (Page 253)

Operation within the print preview

Functions within the print preview

The print preview shows an exact image of the subsequent printout. You can use the buttons in the toolbar to modify the print preview display. The following functions are available (from left to right):

- Navigation mode

  Allows shifting of the partial page.

  To change the partial page with the navigation tool, select the arrow button in the toolbar. Then, click anywhere on the page and hold the mouse button down while moving the page to the desired position.

- Zoom function

  - "Zoom in" and "Zoom out"

    Magnifies or reduces the page display.

    To zoom in or zoom out the display incrementally, select the corresponding button. Then click on the page in order to magnify (zoom in) or reduce (zoom out) the page incrementally.

    To zoom in on a particular area, select the "Lasso zoom" button and use the mouse to drag an outline around the area you want to focus on.

    To select an area to focus on, select the button "Zoom in / zoom out with rectangle". With the mouse, drag a border around the area to focus on it.

    To zoom dynamically through the page, select the button "Zoom in / zoom out dynamically". With pressed mouse button, scroll down over the page to zoom in. Scroll up to zoom out.

  - Percentage value in the drop-down list

    Specifies the display size of the page in percent.

    Enter a percentage value or select a percentage value from the drop-down list. Alternatively, choose the the "Fit to page" option from the drop-down list to adapt the page size to the work area. Or, choose "Fit to width" to adapt the page width to the work area.

- "Forward" and "Backward":

  Each change in the partial page, the page count, or the display size is saved in a history in the background. You can use the "Forward" or "Backward" button to return to the previous view or the next view.

- Page navigation

  - "First page"

    Jumps back to the first page.

  - "Previous page"

    Goes one page back.

  - "Page number" input field

    Shows the current page. To jump directly to a page, enter the page number of the page you want to view.

  - "Next page"

    Goes to the next page.

  - "Last page"

    Jumps to the last page.

See also

Creating a print preview (Page 251)

7.3.4.9 Printing project data

You have two options for printing out project data:

- Print immediately using default settings by means of the "Print" button in the toolbar.

  The button is only active if a printable object is selected.

- Printout with additional setting options with the "Project > Print" menu command.

  For example, you can choose a different printer or specific documentation information or you can specify whether a cover page and table of contents are to be printed. In addition, you can specify the print scope or display a print preview prior to printing.

Requirement

- At least one printer is configured.
- The objects to be printed are not protected.

  If printing is restricted by access protection, you must temporarily cancel the password protection in order to release the objects for printing. Otherwise, protected objects will not be included in the printout.

Printing project data

To print out data from the current project or the entire project with additional setting options, follow these steps:

1. Select the entire project in the project tree in order to print out the entire project. To print only individual elements within a project, select them in the project tree.
2. Select the "Print" command in the "Project" menu.

   The "Print" dialog opens.
3. Select the printer in the "Name" box.
4. Click "Advanced" to modify the Windows printer settings.
5. Select the frame layout you want to use for the printout.
   – Select the documentation information in the "Document information" drop-down list.

     The frame stored in the document information is used for the printout. All placeholders within the chosen frame are filled with the metadata from the selected document information.

   – Select the "Print cover page" check box to print the cover page, which is stored in the selected document information.

   – Select the "Print table of contents" check box to add a table of contents to the printout.

   The check boxes for printing the cover page and the table of contents can only be selected if you have started the printout in the project tree.
6. Under "Print objects/area", select what is to be printed. The selection is only possible if you have started the printout from an editor that supports this function.
   – Choose "All" to print out the entire content of the editor.

   – Choose "Selection" to print only the objects currently selected in the editor.
7. Select the print scope under "Properties".
   – Choose "All" to print all configuration data of the selected objects.

   – Choose "Visible" to print the information of an editor that is currently visible on the screen. This option can only be chosen if you have started the printout from an editor.

   – Choose "Compact" to print out an abbreviated version of the project data.
8. Click "Preview" to generate a print preview in advance.

   A print preview is created in the work area.
9. Click "Print" to start the printout.

Note

Scope of the "Print" dialog

The options available in the "Print" dialog vary depending on the elements to be printed.

Result

The project data is prepared in the background for printing and then printed on the selected printer. The status bar shows the progress of the print operation. You can continue working normally while data is being prepared for printing.

The print results and any errors or warnings are listed in the Inspector window under "Info" at the conclusion of the print job.

Canceling a print job

To cancel an active print job, follow these steps:

1. Click the red "X" in the status bar next to the progress display for the printout.

   The printout is cancelled promptly.

See also

Protection concept for project data (Page 234)

Revoking access rights for devices (Page 235)

Printout of project contents (Page 237)

7.4 Undoing and redoing actions

7.4.1 Basics of undoing and redoing actions

Function

You can undo performed actions at any time. For this purpose, every action you perform is saved in an action stack. When undoing actions, the stack is processed from top to bottom. In other words, if you undo an action that lies further down in the stack, all actions located above it in the stack will also be undone automatically.

You can redo previously undone actions until you execute a new action. Once you execute a new action, it is no longer possible to redo previously undone actions.

Particularities for undoing

There are a few actions that empty the action stack. You cannot undo these actions or the actions performed before these actions. The following actions empty the action stack:

- Saving
- Project management (creating a new project, opening project, closing a project, deleting a project)
- Compiling
- Restoring blocks
- Establishing an online connection
- Loading
- Writing to memory cards

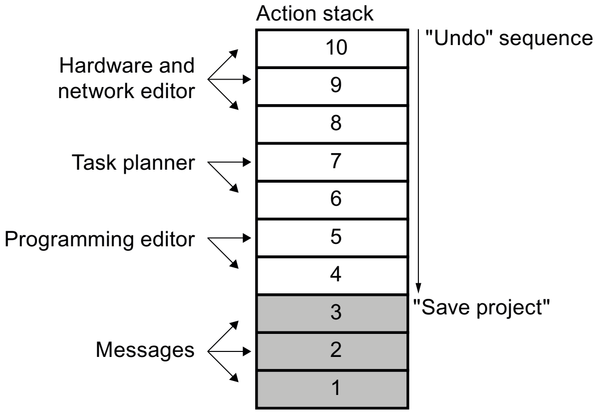
Displaying the action stack

The "Undo" button in the toolbar is enabled as soon as you perform an action that can be undone. This button is split; you can use the arrow down portion to open a drop-down list containing all actions of the action stack that you can undo. If you had performed actions in an editor other than the currently displayed editor, the corresponding editor is also displayed as a subheading. This allows you to always identify the point at which the undo operation will be applied. The subheadings are removed from the list when the editor responsible can no longer undo actions.

Actions you have undone are entered in the action stack from where they can be redone. Here, you can redo actions you have undone. The display of actions you can redo it is analogous to the display of the actions that you can undo.

Example of undoing actions

The figure below shows how actions performed in various editors and tables are undone:



In this example, you cannot undo actions 1 to 3 because the project was saved. You can undo actions 4 to 10 in the order indicated by the direction of the arrow. In other words, you must undo action 10 first. Once you have undone action 8, you cannot then undo action 5. You must first undo actions 7 and 6. As the final step in the sequence, you can then undo action 4. You also have the option of undoing several actions in a single step by undoing an action located further down in the action stack. All actions located above it in the stack will be undone automatically.

The same principle also applies to redoing of actions.

See also

Undoing an action (Page 259)

Redoing an action (Page 260)

7.4.2 Undoing an action

The following options are available for undoing actions:

- Undoing the last action only

  Only the last action performed is undone.
- Undoing as many actions as required

  Multiple actions in the action stack are undone in a single step.

Undoing the last action only

To undo the last action performed, follow the steps below:

1. Click the "Undo" button in the toolbar.
   - If the action was not performed in the currently displayed editor, a confirmation prompt appears.
   - If the undo operation requires an editor containing a protected object to be opened, you must enter the password for the object.
2. Click "Yes" to confirm.
3. Enter the password, if necessary.

   The editor in which the action was performed is displayed and the action is undone.

Undoing as many actions as required

To undo multiple actions in the action stack in a single step, follow these steps:

1. Click the Down arrow next to the "Undo" button in the toolbar.

   This opens a drop-down list containing all actions you can undo. Actions performed in other editors are identified by the editor name in the subheading.
2. Click the action you want to undo.

   The chosen action and all actions in the stack located above the chosen action are undone. If the undo operation requires an editor containing a protected object to be opened, you must enter the password for the object.
3. Enter the required passwords, if necessary.

   The editors in which the actions were performed are displayed and the actions are undone.

See also

Basics of undoing and redoing actions (Page 257)

Redoing an action (Page 260)

7.4.3 Redoing an action

You have the option of redoing an action that has been undone, so that you can return to the status present before "Undo" was performed. However, this is only possible until you perform a new action. The following options are available for redoing actions:

- Redoing the last undone action only

  Only the last undone action is redone.
- Redoing as many undone actions as required

  Multiple undone actions in the action stack are redone in a single step.

Redoing the last undone action only

To redo the last undone action, follow the steps below:

1. Click the "Redo" button in the toolbar.
   - If the action is not being redone in the currently displayed editor, a confirmation prompt appears.
   - If the redo operation requires an editor containing a protected object to be opened, you must enter the password for the object.
2. Click "Yes" to confirm.
3. Enter the password, if necessary.

   The editor in which the action was undone is displayed and the action is redone.

Redoing as many undone actions as required

To redo multiple undone actions in the action stack in a single step, follow these steps:

1. Click the Down arrow next to the "Redo" button in the toolbar.

   This opens a drop-down list containing all actions you can redo. Actions performed in other editors are identified by the editor name in the subheading.
2. Click the action you want to redo.

   The chosen action and all actions in the stack located above the chosen action are redone. If the redo operation requires an editor containing a protected object to be opened, you must enter the password for the object.
3. Enter the required passwords, if necessary.

   The editors in which the actions were undone are displayed and the actions are redone.

See also

Basics of undoing and redoing actions (Page 257)

Undoing an action (Page 259)

7.5 Finding and replacing in projects

7.5.1 Information on the search function

Find and replace

You can search for texts in the editors. The search function finds all texts containing the search key in the currently opened editor. The results are selected in sequence in the opened editor.

You also have the following options:

- Narrowing down the search with additional options
- Replacing found texts

The additional options and the type of texts for which you can search depend on the installed products and the currently open editor.

See also

Search and replace (Page 262)

7.5.2 Search and replace

Using Find

The "Find and replace" function enables you to search for or replace texts in an editor.

Additional options for searching

You can narrow down your search by selecting one of the following additional options:

- Whole words only

  Only whole words are found. Words that contain the search key as part of the word are ignored.

- Match case

  Upper- and lowercase letters are taken into account in the search.

- Find in substructures

  The search also includes texts contained in another object.

- Find in hidden texts

  Texts that are assigned to another text but that are currently hidden are also included in the search.

- Use wildcards

  Enter an asterisk as the wildcard for any number of characters. Example: You want to search for all words starting with "Device". Type in "Device*" in the search key box.

  Enter a question mark as the wildcard, however, if you only want to leave out a single character.

- Use regular expressions (for searching in scripts only)

  A regular expression is a character string used to describe sets of values and for filtering. This allows you to create complex search patterns.

The additional options available depend on the installed products and the editor opened.

Start search

Follow these steps to start the "Find and replace" function:

1. Select the "Find and replace" command in the "Edit" menu or open the "Find and replace" pane in the "Tasks" task card.

   The "Find and replace" pane opens.

2. Enter a term in the "Find" drop-down list.

   As an alternative, you can select the most recent search key from the drop-down list.

3. Select the options desired for the search.

4. Using the option buttons, select the starting point for the search and the search direction.
   – Select "Whole document" if you want to search through the entire editor regardless of the current selection,
   – Select "From current position" if you want to start the search at the current selection.
   – Select "Selection" if you only want to search within the current selection.
   – Select "Down" to search through the editor from top to bottom or from left to right.
   – Select "Up" to search through the editor from bottom to top or from right to left.
5. Click "Find".

   The first hit is marked in the editor.
6. Click "Find" again to display the next hit.

   The next hit is marked in the editor. Repeat this process, as necessary, until you reach the last hit.

Replacing the search key

You have the option of replacing hits individually or automatically replacing all the found texts, if the respective editor supports this function. Follow these steps to replace terms:

1. Enter a term in the "Find" drop-down list.

   As an alternative, you can select the most recent search key from the drop-down list.
2. Select the options desired for the search.
3. Click the "Find" button to start a search for the specified search key.

   The first hit is displayed in the editor.
4. In the "Replace" drop-down list, enter the text you wish to use to replace the search key.

   As an alternative, you can select the most recently text specified from the drop-down list.
5. Click the "Replace" button to replace the selected hit with the specified text.

   The found text is replaced and the next hit is marked in the editor.

   Repeat this process until you have replaced all the hits as wanted. To skip to the next hit without replacing the marked word, click the "Find" button instead of "Replace".
6. Click "Replace all" to automatically replace all hits at once.

See also

Information on the search function (Page 261)

7.6 Working with text lists

7.6.1 Text lists

Introduction

You can manage texts to be referenced in alarms centrally. All the texts are stored in text lists. Each text list has a unique name with which you can call up its content. A range of values is assigned to each text in a text list. If a value from a range of values occurs, the corresponding text is called up.

All the texts can be translated to all project languages. Here, you have two options available:

- You can enter the translation of the texts in a list. You will find the list in the project tree under "Languages & Resources > Project texts".
- You can export all texts to a file in Office Open XML format and enter the translation in a spreadsheet program. The translations can then be imported again.

The texts are translated into the other project languages within the framework of the project texts. In the text lists editor, you only have to manage the assignment of individual texts to a text list.

Each device in the project has its own text lists. For this reason, these lists are arranged under the devices in the project tree. In addition, there are text lists that apply to all devices. These can be found in the project tree under "Common data > Text lists".

User-defined and system-defined text lists

There are two types of text lists:

- User-defined text lists

  You can create user-defined text lists yourself and fill them with texts; in other words, you can specify value ranges and the corresponding texts yourself. With user-defined text lists, the name of the text list begins with "USER" as default. You can change this name to any suitable name.

- System-defined text lists

  System-defined text lists are created by the system. These always involve texts relating to devices. They are automatically created as soon as you insert a device in the project. With system alarms, the name of the text list begins with "SYSTEM". The name of the text list and the ranges of values it contains cannot be modified. You can only edit texts assigned to individual value ranges.

User-defined text lists	System-defined text lists
A user-defined text list can only be assigned to one device.	System-defined text lists can be assigned both to a device as well as to the entire project.
You can create new text lists and delete existing text lists.	You cannot create new text lists or delete text lists.
You can add and delete value ranges in the text lists.	You cannot add or delete value ranges in the text lists.
You can specify both the value ranges as well as the associated texts.	You can only edit the text associated with one value range.

Device-specific and cross-device text lists

Device-specific text lists relate to only one device in the project and are therefore only valid for this device. For this reason, they are arranged under a device in the project tree. Device-specific text lists can be used-defined or created by the system.

If system-defined text lists are generally valid for several devices or not intended uniquely for one device, these are grouped together in the project tree under "Common data". These text lists are available for all devices. Cross-device text lists are always created by the system and are used solely for system diagnostics alarms. For this reason, you cannot store any user-defined text lists under "Common data".

See also

Exporting and importing project texts (Page 220)

7.6.2 Creating user-defined text lists

Creating text lists

You can create user-defined text lists for individual devices.

Requirement

- You are in the project view.
- A project is open.
- The project includes a least one device.

Procedure

To create user-defined text lists, follow these steps:

1. Click on the arrow to the left of a device in the project tree.

   The elements arranged below the device are displayed.
2. Double-click on "Text lists".

   All the text lists assigned to the device are displayed in the work area listed in a table.
3. Double-click on the first free row in the table.

   A new user-defined text list is created.
4. Enter a name for your new text list in the "Name" column.
5. From the drop-down list in the "Selection" column, select whether you want to specify the value ranges in decimal, binary or in bits. Depending on the device, there may be further options available at this point.
6. Enter a comment in the "Comment" column.

   A new user-defined text list has been created and you can now enter the value ranges and texts.

7.6.3 Editing user-defined text lists

Editing user-defined text lists

You can enter value ranges and the corresponding texts in user-defined text lists. User-defined text lists are always located below a device in the project tree.

Requirement

- You are in the project view.
- A project is open.
- The project includes a least one device.

Adding to user-defined text lists with value ranges and texts

To add to user-defined text lists with value ranges and texts, follow these steps:

1. Click on the arrow to the left of a device in the project tree.

   The elements arranged below are displayed.
2. Double-click on "Text lists".

   All the text lists assigned to the device are displayed in the work area listed in a table.
3. Select a text list in the table.

   The contents of the selected text list are displayed in the work area. There, you can enter a value range and assign texts to the individual value ranges.
4. Enter the value ranges you require in the "Range from" and " Range to" columns. The entry must be made in the numeric format selected for the text list.
5. Enter a text for each value range in the "Entry" column.

7.6.4 Editing system-defined text lists

Editing system-defined text lists

In system-defined text lists, you can only modify the individual texts assigned to a value range.

System-defined text lists are located in the project tree either below a device or under "Common data".

Requirement

- You are in the project view.
- A project is open.
- The project includes a least one device.

Modifying texts in system-defined text lists

To edit texts in system-defined text lists that are assigned to a value range, follow these steps:

1. Click on the arrow to the left of a device in the project tree or the "Common data" element.

   The elements arranged below are displayed.

2. Double-click on "Text lists".

   All the text lists assigned to the device or used in common are displayed in the work area listed in a table.

3. Select a text list in the table.

   The contents of the selected text list are displayed in the work area. Here, you can add to or edit the texts assigned to a value range.

4. Enter a text for each value range in the "Entry" column.

7.7 Using memory cards

7.7.1 Basics about memory cards

Introduction

Memory cards are plug-in cards that come in a variety of types and can be used for a variety of purposes. Depending on the device type or device family, memory cards can be used for purposes, such as:

- Load memory of a CPU
- Storage medium for projects, firmware backups, or any other files
- Storage medium for performing a firmware update
- Storage medium for the PROFINET device name

For information regarding the technical variants of the respective memory cards and general information on their handling, refer to the respective documentation for the device. For information on handling memory cards in the TIA Portal, refer to the online help under keyword "Memory Card".

CAUTION
Do not use memory cards for non-SIMATIC-related purposes, and do not use third-party devices or Windows tools to format them. This will irrevocably overwrite the internal structure of the memory card, rendering it unusable for SIMATIC devices!

See also

Adding a user-defined card reader (Page 269)

Accessing memory cards (Page 270)

Displaying properties of memory cards (Page 271)

7.7.2 Adding a user-defined card reader

Introduction

If your card reader is not detected automatically, you can add it manually.

Requirement

The project view is open.

Procedure

To add a card reader, follow these steps:

1. Open the project tree.
2. Select "SIMATIC Card Reader > Add user-defined Card Reader" in the "Project" menu.

   The "Add user-defined Card Reader" dialog opens.
3. In the drop-down list box, select the path for the card reader.
4. Confirm your entries with "OK".

See also

Basics about memory cards (Page 269)

Accessing memory cards (Page 270)

Displaying properties of memory cards (Page 271)

7.7.3 Accessing memory cards

Requirement

- A memory card is inserted in the card reader.
- The project view is open.

Procedure

To access memory cards, follow these steps:

1. Open the project tree.
2. Select "SIMATIC Card Reader > Show SIMATIC Card Reader" in the "Project" menu.

   The "SIMATIC Card Reader" folder is displayed in the project tree.
3. Open the "SIMATIC Card Reader" folder.

   You can now access the memory card.

See also

Basics about memory cards (Page 269)

Adding a user-defined card reader (Page 269)

Displaying properties of memory cards (Page 271)

7.7.4 Displaying properties of memory cards

You can display the properties for the utilized memory cards. Note that different memory cards with different properties must be used, depending on the device.

Requirement

- A memory card is inserted in the card reader.
- The project view is open.

Procedure

To display the properties of a memory card, follow these steps:

1. Right-click on the memory card for which you want to display the properties.
2. Select the "Properties" command in the shortcut menu.

   The "Memory Card <name of the memory card>" dialog opens. The properties are displayed in this dialog.

See also

Basics about memory cards (Page 269)

Adding a user-defined card reader (Page 269)

Accessing memory cards (Page 270)

7.8 Using libraries

7.8.1 Library basics

Introduction

You can store objects you want to use more than once in libraries. It is possible to reuse the stored objects throughout a project or across projects. This means you can, for example, create block templates for use in different projects and adapt them to the particular requirements of your automation task.

Library types

Depending on the task, you can use one of the following library types:

- Project library

  Each project has its own library. Here, you store the objects you want to use more than once in the project. This project library is always opened, saved, and closed together with the current project.

- Global libraries

  You can also create other libraries in addition to the project library. Here, you store the objects you want to use in more than one project. You can create, change, save, and transfer these global libraries independent of projects.

  In the global libraries area, you will also find libraries that ship with the software. These include off-the-peg functions and function blocks that you can use within your project. You cannot modify the supplied libraries.

Library objects

Libraries can accommodate a large number of objects. These include, for example:

- Functions (FCs)
- Function blocks (FBs)
- Data blocks (DBs)
- Devices
- PLC data types
- Watch and force tables
- Process screens
- Faceplates

For objects with know-how protection, this protection is also retained after the object is inserted into a library.

Use types

You can use the library elements as either a copy template or a type. You can use copy templates to generate copies of the library element that are independent of one another. Copy templates are marked with a black triangle in the "Libraries" task card. You can derive and use instances from types. These instances are tied to their respective type, and changes to one instance also change all other instances. Types are marked with a green triangle in the "Libraries" task card.

Note

Please note the following:

- The use as a type is not available for every object.
- You can only create types in the project library.

See also

Project library basics (Page 276)

Global library basics (Page 285)

"Libraries" task card (Page 273)

7.8.2 "Libraries" task card

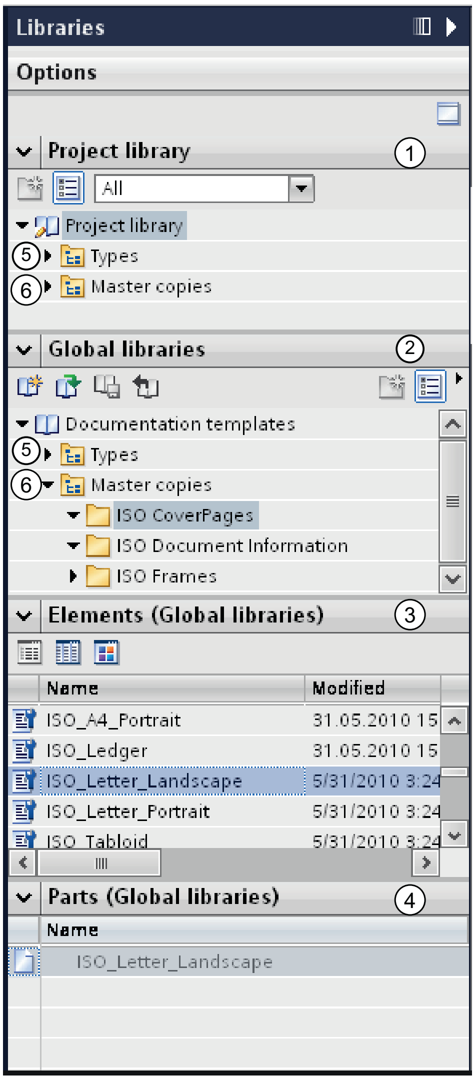
Function of the "Libraries" task card

The "Libraries" task card enables you to work efficiently with the project library and the global libraries.

You can show or hide the task card as needed.

Structure of the "Libraries" task card

The "Libraries" task card consists of the following components:



Name	Modified
ISO_A4_Portrait	31.05.2010 15
ISO_Ledger	31.05.2010 15
ISO_Letter_Landscape	5/31/2010 3:24
ISO_Letter_Portrait	5/31/2010 3:24
ISO_Tabloid	5/31/2010 3:24

① "Project library" pane

② "Global libraries" pane

③ "Elements" pane

④ "Parts" pane

⑤ "Types" folder

⑥ "Master copies" folder

"Project library" pane

In this pane, you can store the objects that you want to use more than once in the project.

"Global libraries" pane

In this pane, you can store the objects you want to use more than once in various projects.

The "Global libraries" pane also lists libraries that ship with the system. These libraries provide you with ready-made functions and function blocks, for example. You can use these supplied global libraries but cannot modify them.

"Elements" pane

In this pane, you can display the elements of a library.

"Parts" pane

In this pane, you can display the contents of the library elements.

"Types" folder

In this directory, you can create types of your objects in the project library and insert them as instances. You cannot create types in a global library. You can copy a type in the project library and paste it into the "Types" folder of a global library.

"Master copies" folder

In this directory, you can create copy templates of your objects, which you can insert as copies.

See also

Library basics (Page 272)

Using the elements and parts view (Page 276)

7.8.3 Using the elements and parts view

Introduction

When you open the "Libraries" task card the first time, the "Project library" and "Global libraries" panes are opened and the "Parts" pane is closed. You need to open the "Elements" pane explicitly.

The elements view shows the elements of the selected library. You can select one of the following views:

- Details
- List
- Overview

The parts view shows the contents of the selected library element.

Requirement

The "Libraries" task card is displayed.

Procedure

To use the elements and parts view, follow these steps:

1. Click "Open or close the element view" in the "Project library" or "Global libraries" pane.

See also

Library basics (Page 272)

"Libraries" task card (Page 273)

7.8.4 Working with the project library

7.8.4.1 Project library basics

Function

In the project library, you can store the elements that you want to use more than once in the project. The project library is generated and saved automatically with the project.

See also

Library basics (Page 272)

7.8.4.2 Creating folders in the project library

The library elements are stored according to their type in the "Types" and "Master copies" folders within the project library. You can add subfolders to these folders as required.

Requirement

The "Libraries" task card is displayed.

Procedure

To create a new folder, follow these steps:

1. Right-click any folder within the project library.
2. Select "Add folder" from the shortcut menu.

   A new folder is created.
3. Enter a name for the new folder.

See also

Project library basics (Page 276)

Editing elements of a project library (Page 282)

Removing elements from the project library (Page 284)

Filtering the view (Page 285)

7.8.4.3 Adding elements to the project library

Adding master copies to the project library

Requirement

The "Libraries" task card is displayed.

Procedure

To add a new copy template to the project library, follow these steps:

1. Open the project library in the "Project library" pane of the "Libraries" task card.
2. Select the element you want to add as a copy template to the project library, and use a drag-and-drop operation to move it to the "Master copies" folder or any of its subfolders in the project library. Release the left mouse button when a small plus symbol appears below the mouse pointer.

Result

The element is inserted into the project library as a copy template. You can generate copies from this template and use them anywhere in the TIA Portal where it is permissible.

See also

Library basics (Page 272)

Project library basics (Page 276)

Adding types to the project library (Page 278)

Adding types to the project library

Requirement

- Your installed products contain objects from which types can be generated.
- The "Libraries" task card is displayed.

Procedure

To add a new type to the project library, follow these steps:

1. Open the project library in the "Project library" pane of the "Libraries" task card.
2. Right-click the "Types" folder or any of its subfolders.
3. Select the "Add new type" command in the shortcut menu.

   The dialog for generating types opens.

   Note

   If your installed products do not contain any objects from which types can be generated, the "Add new type" command is not available. Use master copies in this case.

4. In the dialog, choose the element from which you want to generate a type.
5. Specify all other necessary data.
6. Click "OK".

Result

A new type is generated and inserted into the project library. You can generate instances from this type and use them anywhere in the TIA Portal where it is permissible. The instances are tied to the type, i.e., changes to one instance are passed on to the other instances.

See also

Library basics (Page 272)

Project library basics (Page 276)

Adding master copies to the project library (Page 277)

7.8.4.4 Using elements of the project library

Using master copies

You use the master copies you inserted into the project library in order to generate copies of elements and insert them in the TIA Portal, where permitted. The copies are generated and used in a single operation.

Requirement

The "Libraries" task card is displayed.

Procedure

To generate and use a copy from a copy template, follow these steps:

1. Open the "Master copies" folder or any of its subfolders in the project library so that you looking at the master copy from which you want to generate a copy.
2. Use a drag-and-drop operation to move the copy template from the project library to the location where you want to use it. If you are not permitted to insert it at this location, the mouse pointer changes to a circle with a slash.

   A copy based on the copy template is inserted. If there is already an element with the same name at this location, the "Paste" dialog opens. In this dialog, you can decide whether to replace the existing element or to insert the copy with a different name.

   Note

   The "Paste" dialog is not displayed for all elements. The elements for which the dialog is available depends on the installed products.

Or:

1. Open the element view.
2. Use a drag-and-drop operation to move the copy template from the "Elements" pane to the location where you want to use it. If you are not permitted to insert it at this location, the mouse pointer changes to a circle with a slash.

   A copy based on the copy template is inserted. If there is already an element with the same name at this location, the "Paste" dialog opens. In this dialog, you can decide whether to replace the existing element or to insert the copy with a different name.

   Note

   The "Paste" dialog is not displayed for all elements. The elements for which the dialog is available depends on the installed products.

Result

A copy is generated from the copy template and inserted at the location where you want to use it. You can create any number of copies from a copy template.

See also

Library basics (Page 272)

Project library basics (Page 276)

Using types (Page 281)

Using the elements and parts view (Page 276)

Using types

You use the types you inserted into the project library in order to generate instances of elements and insert them in the TIA Portal, where permissible. The instances are generated and used in a single operation.

Requirement

The "Libraries" task card is displayed.

Procedure

To generate and use an instance from a type, follow these steps:

1. Open the "Types" folder or any of its subfolders in the project library so that you looking at the type from which you want to generate an instance.
2. Use a drag-and-drop operation to move the type from the project library to the location where you want to use it. If you are not permitted to insert it at this location, the mouse pointer changes to a circle with a slash.

   The type is inserted as an instance. If there is already an element with the same name at this location, the "Paste" dialog opens. In this dialog, you can decide whether to replace the existing element or to insert the instance with a different name.

   ### Note

   The "Paste" dialog is not displayed for all elements. The elements for which the dialog is available depends on the installed products.

Or:

1. Open the element view.
2. Use a drag-and-drop operation to move the type from the "Elements" pane to the location where you want to use it. If you are not permitted to insert it at this location, the mouse pointer changes to a circle with a slash.

   The type is inserted as an instance. If there is already an element with the same name at this location, the "Paste" dialog opens. In this dialog, you can decide whether to replace the existing element or to insert the instance with a different name.

   ### Note

   The "Paste" dialog is not displayed for all elements. The elements for which the dialog is available depends on the installed products.

Result

An instance is generated from the type and inserted at the location where you want to use it. You can create any number of instances from a type. The instances are tied to the type, i.e., changes to one instance are passed on to the other instances.

See also

Library basics (Page 272)

Project library basics (Page 276)

Using master copies (Page 279)

7.8.4.5 Editing elements of a project library

You can use the following editing commands on library elements:

- Copy
- Cut
- Paste
- Move within the library
- Rename

Each of these commands can always be executed via the keyboard (Page 194), menu, and shortcut menu.

Requirement

The "Libraries" task card is displayed.

Copying elements

To copy a library element, follow these steps:

1. Right-click on the library element you want to copy.
2. Select the "Copy" command in the shortcut menu.

Cutting elements

To cut a library element, follow these steps:

1. Right-click on the library element you want to cut.
2. Select the "Cut" command in the shortcut menu.

Note

You can only paste previously cut library elements into the same library. In so doing, you can only paste master copies into the "Master copies" folder or any of its subfolders. Likewise, you can only paste types into the "Types" folder or any of its subfolders.

Pasting elements

To paste a library element, follow these steps:

1. Copy a library element.
2. Right-click the library where you want to paste the element.
3. Select the "Paste" command in the shortcut menu.

Moving elements

To move a library element, follow these steps:

1. Select the library element you want to move.
2. Use a drag-and-drop operation to move the library element to the folder where you want to insert the element.

Note

Note the following:

- When you move an element from one library to another, the element is copied and not moved.
- You cannot move master copies into a type folder or a type into a master copies folder.

Renaming elements

To rename a library element, follow these steps:

1. Right-click the element you want to rename.
2. Select the "Rename" command in the shortcut menu.
3. Enter the new name.

See also

Library basics (Page 272)

Project library basics (Page 276)

Creating folders in the project library (Page 277)

Removing elements from the project library (Page 284)

Filtering the view (Page 285)

7.8.4.6 Removing elements from the project library

Requirements

The "Libraries" task card is displayed.

Procedure

To remove an element from the project library, follow these steps:

1. In the "Project library" pane, maximize the folder containing the element you want to remove.
2. Right-click the element.
3. Select the "Delete" command in the shortcut menu.

Or:

1. Open the element view.
2. Right-click the element you want to remove in the "Elements" pane.
3. Select the "Delete" command in the shortcut menu.

See also

Library basics (Page 272)

Project library basics (Page 276)

Creating folders in the project library (Page 277)

Editing elements of a project library (Page 282)

Filtering the view (Page 285)

7.8.4.7 Filtering the view

To make viewing of extensive libraries more straightforward, you can use filter options to limit the display.

Requirement

The "Libraries" task card is displayed.

Procedure

To filter the view, follow these steps:

1. Open either the "Project library" pane or "Global libraries" pane.
2. In the drop-down list, select the object type for which you want to display the library elements.

Result

Only the library elements that are available for the object type are displayed. You can set the filter to "All" at any time to revert to an unfiltered view.

See also

Library basics (Page 272)

Project library basics (Page 276)

Creating folders in the project library (Page 277)

Editing elements of a project library (Page 282)

Removing elements from the project library (Page 284)

7.8.5 Working with global libraries

7.8.5.1 Global library basics

Function

You can store elements that you want to reuse in other projects in global libraries. You must create global libraries explicitly.

Depending on the products installed, global libraries supplied by Siemens are also installed. You can use, but not change, the elements of these libraries.

Shared work with global libraries

You can use global libraries together with other users. This requires, however, that all users who want to access to the global library open the library as read-only.

See also

Library basics (Page 272)

7.8.5.2 Creating a new global library

Requirement

The "Libraries" task card is displayed.

Procedure

To create a new global library, follow these steps:

1. Click "Create new global library" in the toolbar of the "Global libraries" pane or select the menu command "Options > Global libraries > Create new library".

   The "Create new global library" dialog opens.
2. Specify the name and the storage location for the new global library.
3. Confirm your entries with "Create".

Result

The new global library is generated and inserted into the "Global libraries" pane. A folder with the name of the global library is created in the file system at the storage location of the global library. This actual library file is given the file name extension ".al11".

See also

Library basics (Page 272)

Global library basics (Page 285)

Opening a global library (Page 287)

Displaying properties of global libraries (Page 288)

Saving a global library (Page 289)

Closing a global library (Page 290)

Deleting a global library (Page 291)

Creating folders in the global libraries (Page 292)

Editing elements of a global library (Page 298)

Removing elements from a global library (Page 300)

Using a supplied global library (Page 301)

Filtering the view (Page 302)

7.8.5.3 Opening a global library

Requirement

The "Libraries" task card is displayed.

Procedure

To open a global library, follow these steps:

1. Click "Open global library" in the toolbar of the "Global libraries" pane or select the menu command "Options > Global libraries > Open library".

   The "Open global library" dialog box is displayed.
2. Select the global library you want to open. Library files are identified by the file name extension ".al11".
3. You can also open the library so that it is write-protected. To do this, enable the "Open as read-only" option in the "Open global library" dialog.

   Note

   Note the following:

   - You cannot enter any additional elements in the global library if you open the library as read-only.
   - All users have to open the library as read-only if multiple users want to access the library. This is a requirement for shared access to the library.
4. Click "Open".

   The selected global library is opened and inserted into the "Global libraries" pane.

See also

Library basics (Page 272)

Global library basics (Page 285)

Creating a new global library (Page 286)

Displaying properties of global libraries (Page 288)

Saving a global library (Page 289)

Closing a global library (Page 290)

Deleting a global library (Page 291)

Creating folders in the global libraries (Page 292)

Editing elements of a global library (Page 298)

Removing elements from a global library (Page 300)

Using a supplied global library (Page 301)

Filtering the view (Page 302)

7.8.5.4 Displaying properties of global libraries

You can display the properties of global libraries. Properties include the following:

- General information about the library

  This includes the following information: creation time, author, file path, file size, copyright, etc. Many of the attributes can be changed.
- Library history

  The library history contains an overview of the migrations performed. Here you can also call the log file for the migrations.
- Support packages in the library

  You can display an overview of the additional software needed to work with all devices in the project.
- Software products in the library

  You can display an overview of all installed software products needed for the project.

Requirement

The "Libraries" task card is displayed.

Procedure

To display the properties of a global library, follow these steps:

1. Right-click the global library whose properties you want to display.
2. Select the "Properties" command in the shortcut menu.

   A dialog containing the properties of the global libraries opens.
3. Select the properties in the area navigation that you want to have displayed.

See also

Opening a global library (Page 287)

Library basics (Page 272)

Global library basics (Page 285)

Creating a new global library (Page 286)

Saving a global library (Page 289)

Closing a global library (Page 290)

Deleting a global library (Page 291)

Creating folders in the global libraries (Page 292)

Editing elements of a global library (Page 298)

Removing elements from a global library (Page 300)

Using a supplied global library (Page 301)

Filtering the view (Page 302)

7.8.5.5 Saving a global library

You can save changes made to global libraries not supplied by Siemens at any time. You can save a global library under another name using the "Save library as" command.

Requirement

- The "Libraries" task card is displayed.
- The global library is not write protected.

Saving changes

To save a global library, follow these steps:

1. Right-click on the global library you want to save.
2. Select the "Save library" command in the shortcut menu.

Saving a global library under another name

To save a global library under another name, follow these steps:

1. Right-click the global library that you want to save under a different name.
2. Select the "Save library as" command in the shortcut menu.

   The "Save global library as" dialog opens.
3. Select the storage location and enter the file name.
4. Confirm your entries with "Save".

   The library is saved in the specified location under the new name. The original library is retained.

See also

Library basics (Page 272)

Global library basics (Page 285)

Creating a new global library (Page 286)

Opening a global library (Page 287)

Displaying properties of global libraries (Page 288)

Closing a global library (Page 290)

Deleting a global library (Page 291)

Creating folders in the global libraries (Page 292)

Editing elements of a global library (Page 298)

Removing elements from a global library (Page 300)

Using a supplied global library (Page 301)

Filtering the view (Page 302)

7.8.5.6 Closing a global library

Global libraries are independent of projects. This means that they are not closed together with your project. You must therefore close global libraries explicitly.

Requirement

The "Libraries" task card is displayed.

Procedure

To close a global library, follow these steps:

1. Right-click on the global library you want to close.
2. Select the "Close library" command in the shortcut menu.
3. If you make changes to the global library, a dialog box opens where you can choose whether you want to save the changes to the global library. Click "Yes" or "No", depending on whether or not you would like to save your changes.

   The global library is closed.

See also

Creating a new global library (Page 286)

Opening a global library (Page 287)

Displaying properties of global libraries (Page 288)

Saving a global library (Page 289)

Library basics (Page 272)

Global library basics (Page 285)

Deleting a global library (Page 291)

Creating folders in the global libraries (Page 292)

Editing elements of a global library (Page 298)

Removing elements from a global library (Page 300)

Using a supplied global library (Page 301)

Filtering the view (Page 302)

7.8.5.7 Deleting a global library

You can delete global libraries that were not supplied by Siemens. Note, however, that this action causes deletion of the entire library directory in the file system of your programming device or PC.

Requirements

The "Libraries" task card is displayed.

Procedure

To delete a global library, follow these steps:

1. Right-click the global library you want to delete.
2. Select the "Delete" command in the shortcut menu.
3. Click "Yes" to confirm.

Result

The global library is removed from the "Global libraries" pane and deleted from the file system.

See also

Library basics (Page 272)

Global library basics (Page 285)

Creating a new global library (Page 286)

Opening a global library (Page 287)

Displaying properties of global libraries (Page 288)

Saving a global library (Page 289)

Closing a global library (Page 290)

Creating folders in the global libraries (Page 292)

Editing elements of a global library (Page 298)

Removing elements from a global library (Page 300)

Using a supplied global library (Page 301)

Filtering the view (Page 302)

7.8.5.8 Creating folders in the global libraries

The library elements are stored according to their type in the "Types" and "Master copies" folders within global libraries. You can add subfolders to these folders as required.

Requirements

- The "Libraries" task card is displayed.
- The global library is not write protected.

Procedure

To create a new folder, follow these steps:

1. Right-click any folder within the global library.
2. Select "Add folder" from the shortcut menu.

   A new folder is created.
3. Enter a name for the new folder.

See also

Global library basics (Page 285)

Library basics (Page 272)

Creating a new global library (Page 286)

Opening a global library (Page 287)

Displaying properties of global libraries (Page 288)

Saving a global library (Page 289)

Closing a global library (Page 290)

Deleting a global library (Page 291)

Editing elements of a global library (Page 298)

Removing elements from a global library (Page 300)

Using a supplied global library (Page 301)

Filtering the view (Page 302)

7.8.5.9 Adding elements to a global library

Adding master copies to a global library

Requirements

- The "Libraries" task card is displayed.
- The global library is not write protected.

Procedure

To add a new copy template to a global library, follow these steps:

1. In the Global libraries" pane of the "Libraries" task card, open the global library to which you want to add the copy template.
2. Select the element you want to add as a master copy to the global library, and use a drag-and-drop operation to move it to the "Master copies" folder or any of its subfolders in the global library. Release the left mouse button when a small plus symbol appears below the mouse pointer.

Result

The element is inserted into the global library as a copy template. You can generate copies from this template and use them anywhere in the TIA Portal where it is permissible.

See also

Library basics (Page 272)

Global library basics (Page 285)

Adding types to a global library (Page 294)

Adding types to a global library

You can paste an existing type into a global library type from the project library. This allows you to use this type across multiple projects.

Requirement

- The "Libraries" task card is displayed.
- A type is available in the project library.

Procedure

To add a type to a global library, follow these steps:

1. Open the project library in the "Project library" pane of the "Libraries" task card.
2. Right-click the type which you want to paste into the global library.
3. In the shortcut menu, select "Copy".
4. Right-click the type which you want to paste into the global library.
5. In the shortcut menu, select "Paste".

Result

The type is inserted in the global library. You can generate instances from this type and use them anywhere in the TIA Portal where it is permissible. The instances are tied to the type, i.e., changes to one instance are passed on to the other instances.

See also

Library basics (Page 272)

Global library basics (Page 285)

Adding master copies to a global library (Page 293)

7.8.5.10 Using elements of a global library

Using master copies

You use the master copies you inserted into the global library in order to generate copies of elements and insert them in the TIA Portal, where permitted. The copies are generated and used in a single operation.

Requirements

The "Libraries" task card is displayed.

Procedure

To generate and use a copy from a copy template, follow these steps:

1. Open the "Master copies" folder or any of its subfolders in the global library so that you looking at the master copy from which you want to generate a copy.
2. Use a drag-and-drop operation to move the copy template from the global library to the location where you want to use it. If you are not permitted to insert it at this location, the mouse pointer changes to a circle with a slash.

   A copy based on the copy template is inserted. If there is already an element with the same name at this location, the "Paste" dialog opens. In this dialog, you can decide whether to replace the existing element or to insert the copy with a different name.

   Note

   The "Paste" dialog is not displayed for all elements. The elements for which the dialog is available depends on the installed products.

Or:

1. Open the element view.
2. Use a drag-and-drop operation to move the copy template from the "Elements" pane or the "Parts" pane to the location where you want to use it. If you are not permitted to insert it at this location, the mouse pointer changes to a circle with a slash.

   A copy based on the copy template is inserted. If there is already an element with the same name at this location, the "Paste" dialog opens. In this dialog, you can decide whether to replace the existing element or to insert the copy with a different name.

   Note

   The "Paste" dialog is not displayed for all elements. The elements for which the dialog is available depends on the installed products.

Result

A copy is generated from the copy template and inserted at the location where you want to use it. You can create any number of copies from a copy template.

See also

Library basics (Page 272)

Global library basics (Page 285)

Using types (Page 296)

Using the elements and parts view (Page 276)

Using types

You use the types you inserted into a global library in order to generate instances of elements and insert them in the TIA Portal, where permissible. The instances are generated and used in a single operation.

Requirements

The "Libraries" task card is displayed.

Procedure

To generate and use an instance from a type, follow these steps:

1. Open the "Types" folder or any of its subfolders in the global library so that you looking at the type from which you want to generate an instance.
2. Use a drag-and-drop operation to move the type from the global library to the location where you want to use it. If you are not permitted to insert it at this location, the mouse pointer changes to a circle with a slash.

   The type is inserted as an instance. If there is already an element with the same name at this location, the "Paste" dialog opens. In this dialog, you can decide whether to replace the existing element or to insert the instance with a different name.

   Note

   The "Paste" dialog is not displayed for all elements. The elements for which the dialog is available depends on the installed products.

Or:

1. Open the element view.
2. Use a drag-and-drop operation to move the type from the "Elements" pane to the location where you want to use it. If you are not permitted to insert it at this location, the mouse pointer changes to a circle with a slash.

   The type is inserted as an instance. If there is already an element with the same name at this location, the "Paste" dialog opens. In this dialog, you can decide whether to replace the existing element or to insert the instance with a different name.

   Note

   The "Paste" dialog is not displayed for all elements. The elements for which the dialog is available depends on the installed products.

Result

An instance is generated from the type and inserted at the location where you want to use it. You can create any number of instances from a type. The instances are tied to the type, i.e., changes to one instance are passed on to the other instances.

See also

Library basics (Page 272)

Global library basics (Page 285)

Using master copies (Page 295)

Using the elements and parts view (Page 276)

7.8.5.11 Editing elements of a global library

You can use the following editing commands on library elements:

- Copy
- Cut
- Paste
- Move within the library
- Rename

Each of these commands can always be executed via the keyboard (Page 194), menu, and shortcut menu.

Requirements

- The "Libraries" task card is displayed.
- The global library is not write protected.

Copying elements

To copy a library element, follow these steps:

1. Right-click on the library element you want to copy.
2. Select the "Copy" command in the shortcut menu.

Cutting elements

To cut a library element, follow these steps:

1. Right-click on the library element you want to cut.
2. Select the "Cut" command in the shortcut menu.

Note

You can only paste previously cut library elements into the same library. In so doing, you can only paste master copies into the "Master copies" folder or any of its subfolders. Likewise, you can only paste types into the "Types" folder or any of its subfolders.

Pasting elements

To paste a library element, follow these steps:

1. Copy a library element.
2. Right-click the library where you want to paste the element.
3. Select the "Paste" command in the shortcut menu.

Moving elements

To move a library element within a library, follow these steps:

1. Select the library element you want to move.
2. Drag the library element to the library where you want to insert the element.

Note

When you move an element from one library to another, the element is copied and not moved.

Renaming elements

To rename a library element, follow these steps:

1. Right-click the element you want to rename.
2. Select the "Rename" command in the shortcut menu.
3. Enter the new name.

See also

Library basics (Page 272)

Global library basics (Page 285)

Creating a new global library (Page 286)

Opening a global library (Page 287)

Displaying properties of global libraries (Page 288)

Saving a global library (Page 289)

Closing a global library (Page 290)

Deleting a global library (Page 291)

Creating folders in the global libraries (Page 292)

Removing elements from a global library (Page 300)

Using a supplied global library (Page 301)

Filtering the view (Page 302)

7.8.5.12 Removing elements from a global library

Requirements

- The "Libraries" task card is displayed.
- The global library is not write protected.

Procedure

To remove an element from a global library, follow these steps:

1. In the "Global libraries" pane, maximize the folder containing the element you want to remove.
2. Right-click the element.
3. Select the "Delete" command in the shortcut menu.

Or:

1. Open the element view.
2. Right-click the element you want to remove in the "Elements" pane.
3. Select the "Delete" command in the shortcut menu.

See also

Library basics (Page 272)

Global library basics (Page 285)

Creating a new global library (Page 286)

Opening a global library (Page 287)

Displaying properties of global libraries (Page 288)

Saving a global library (Page 289)

Closing a global library (Page 290)

Deleting a global library (Page 291)

Creating folders in the global libraries (Page 292)

Editing elements of a global library (Page 298)

Using a supplied global library (Page 301)

Filtering the view (Page 302)

7.8.5.13 Using a supplied global library

Depending on the products you install, various global libraries ship with the system.

Requirement

The "Libraries" task card is displayed.

Procedure

To use an element from a supplied global library in your project, follow these steps:

1. Open the relevant library so that you can see the elements of the library.
2. Drag the element from the "Global libraries" pane and drop it on the location where you want to use it. If you are not permitted to insert it at this location, the mouse pointer changes to a circle with a slash.

Or:

1. Open the element view.
2. Use a drag-and-drop operation to move the element from the "Parts" pane to the location where you want to use it. If you are not permitted to insert it at this location, the mouse pointer changes to a circle with a slash.

See also

Library basics (Page 272)

Global library basics (Page 285)

Creating a new global library (Page 286)

Opening a global library (Page 287)

Displaying properties of global libraries (Page 288)

Saving a global library (Page 289)

Closing a global library (Page 290)

Deleting a global library (Page 291)

Creating folders in the global libraries (Page 292)

Editing elements of a global library (Page 298)

Removing elements from a global library (Page 300)

Filtering the view (Page 302)

7.8.5.14 Filtering the view

To make viewing of extensive libraries more straightforward, you can use filter options to limit the display.

Requirement

The "Libraries" task card is displayed.

Procedure

To filter the view, follow these steps:

1. Open either the "Project library" pane or "Global libraries" pane.
2. In the drop-down list, select the object type for which you want to display the library elements.

Result

Only the library elements that are available for the object type are displayed. You can set the filter to "All" at any time to revert to an unfiltered view.

See also

Library basics (Page 272)

Global library basics (Page 285)

Creating a new global library (Page 286)

Opening a global library (Page 287)

Displaying properties of global libraries (Page 288)

Saving a global library (Page 289)

Closing a global library (Page 290)

Deleting a global library (Page 291)

Creating folders in the global libraries (Page 292)

Editing elements of a global library (Page 298)

Removing elements from a global library (Page 300)

Using a supplied global library (Page 301)

7.9 Using cross-references

7.9.1 Using cross-references

Introduction to cross-references

The cross-reference list provides an overview of the use of objects within the project. You can see which objects are interdependent and where the individual objects are located. Cross-references are therefore part of the project documentation.

You can also jump directly to the point of use of an object.

Which objects you can display and localize in the cross-reference list depends on the installed products.

7.10 Simulating devices

7.10.1 Simulation of devices

Introduction

You can use the TIA Portal to run and test the hardware and software of the project in a simulated environment. The simulation is performed directly on the programming device or PC. No additional hardware is required.

The simulation software provides a graphical user interface for monitoring and changing the configuration. It differs according to the currently selected device.

Integration in the TIA Portal

The simulation software is fully integrated in the TIA Portal but is only supported by certain devices. Therefore, the button for calling the simulation software is only active if the selected device supports simulation.

The simulation software for some devices requires its own virtual interface to communicate with the simulated devices. The virtual interface can be found in the project tree under the "Online access" entry next to the physical interfaces of the programming device/PC.

Once you have opened the software, additional help on the simulation software is available via a separate link.

See also

Starting the simulation (Page 304)

7.10.2 Starting the simulation

Some devices can be simulated with additional software. You therefore do not have to have the actual devices to perform comprehensive testing of your project.

Procedure

To start the simulation software, follow these steps:

1. Select the device you want to simulate, for example, in the project tree.
2. Select the "Simulation > Start" command in the "Online" menu.

   This calls the simulation software.

See also

Simulation of devices (Page 304)

Editing devices and networks 8

8.1 Configuring devices and networks

8.1.1 Hardware and network editor

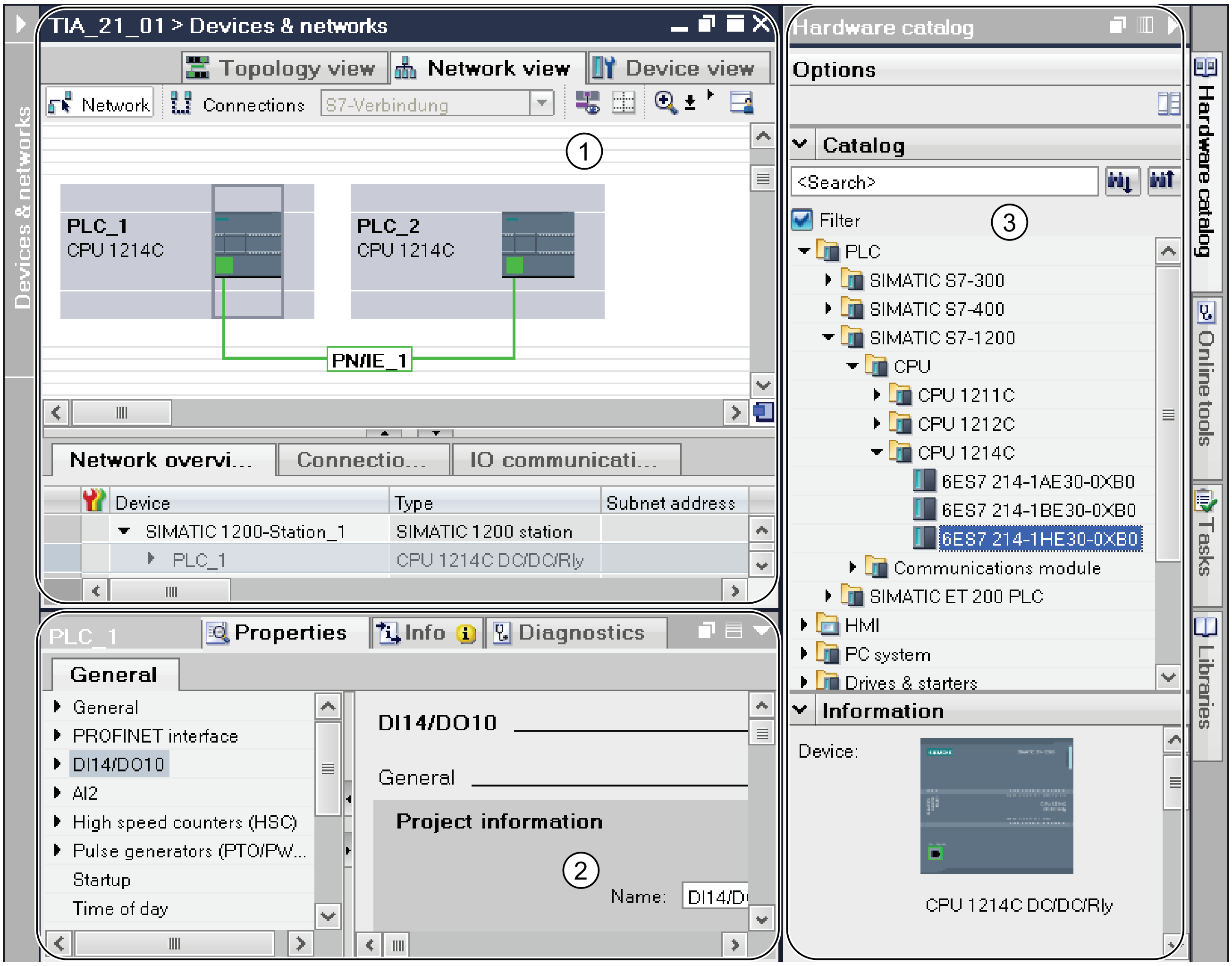
8.1.1.1 Overview of hardware and network editor

Function of the hardware and network editor

The hardware and network editor opens when you double-click on the "Devices and Networks" entry in the project tree. The hardware and network editor is the integrated development environment for configuring, networking and assigning parameters to devices and modules. It provides maximum support for the realization of the automation project.

Structure of the hardware and network editor

The hardware and network editor consists of the following components:



Device	Type	Subnet address
SIMATIC 1200-Station_1	SIMATIC 1200 station	
PLC_1	CPU 1214C DC/DC/Rly	

Name: DI14/Di

① Device view (Page 310), network view (Page 307), topology view (Page 313)

② Inspector window (Page 318)

③ Hardware catalog (Page 320)

The hardware and network editor provides you with three views of your project. You can switch between these three views at any time depending on whether you want to produce and edit individual devices and modules, entire networks and device configurations or the topological structure of your project.

The inspector window contains information on the object currently marked. Here you can change the settings for the object marked.

Drag the devices and modules you need for your automation system from the hardware catalog to the network, device ot topology view.

8.1.1.2 Network view

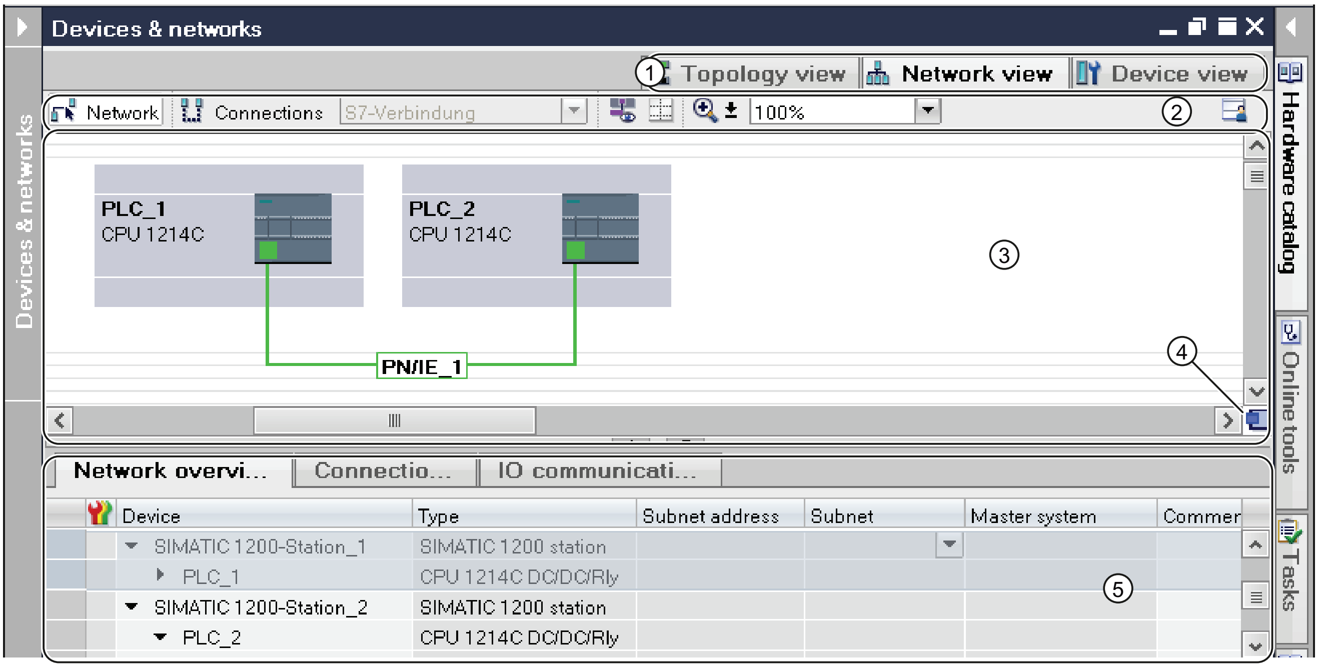
Introduction

The network view is one of three working areas of the hardware and network editor. You undertake the following tasks here:

- Configuring and assign device parameters
- Networking devices with one another

Structure

The following diagram shows the components of the network view:



Device	Type	Subnet address	Subnet	Master system	Commer
SIMATIC 1200-Station_1	SIMATIC 1200 station				
PLC_1	CPU 1214C DC/DC/Rly				
SIMATIC 1200-Station_2	SIMATIC 1200 station				
PLC_2	CPU 1214C DC/DC/Rly				

① Changeover switch: device view / network view / topology view
② Toolbar of network view
③ Graphic area of network view
④ Overview navigation
⑤ Table area of network view

You can use your mouse to change the spacing between the graphic and table areas of the network view. To do this, click on the upper edge of the table view and expand or contract this by moving the mouse with the mouse button held down. You can use the two small arrow keys to minimize, maximize or select the latest table structure of the table view with just one click.

Toolbar

The toolbar provides the following functions:

Symbol	Meaning
	Mode to network devices.
	Mode to create connections. You can define the connection type using the drop-down list.
	Mode to create relations.
	Show interface addresses.
🔍±	Adjust the zoom setting. You can select the zoom setting or enter it directly in the drop-down list. You can also zoom in or zoom out the view in steps using the zoom symbol or draw a frame around an area to be zoomed in.
▦	Show page breaks Enables page break preview. Dotted lines will be displayed at the positions where the pages will break when printed.
	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the network view displays any network-related devices, networks, connections and relations. In this area, you add devices from the hardware catalog, connect them with each other via their interfaces and configure the communication settings.

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate to the desired objects and display them in the graphic area.

Table area

The table area of the network view includes various tables for the devices, connections and communication settings present:

- Network overview
- Connections
- I/O communication

You can use the shortcut menu of the title bar of the table to adjust the tabular display.

See also

Adding a device to the hardware configuration (Page 336)

Determination of online status and display using symbols (Page 649)

Networking devices in the network view (Page 349)

Tabular network overview (Page 352)

8.1.1.3 Device view

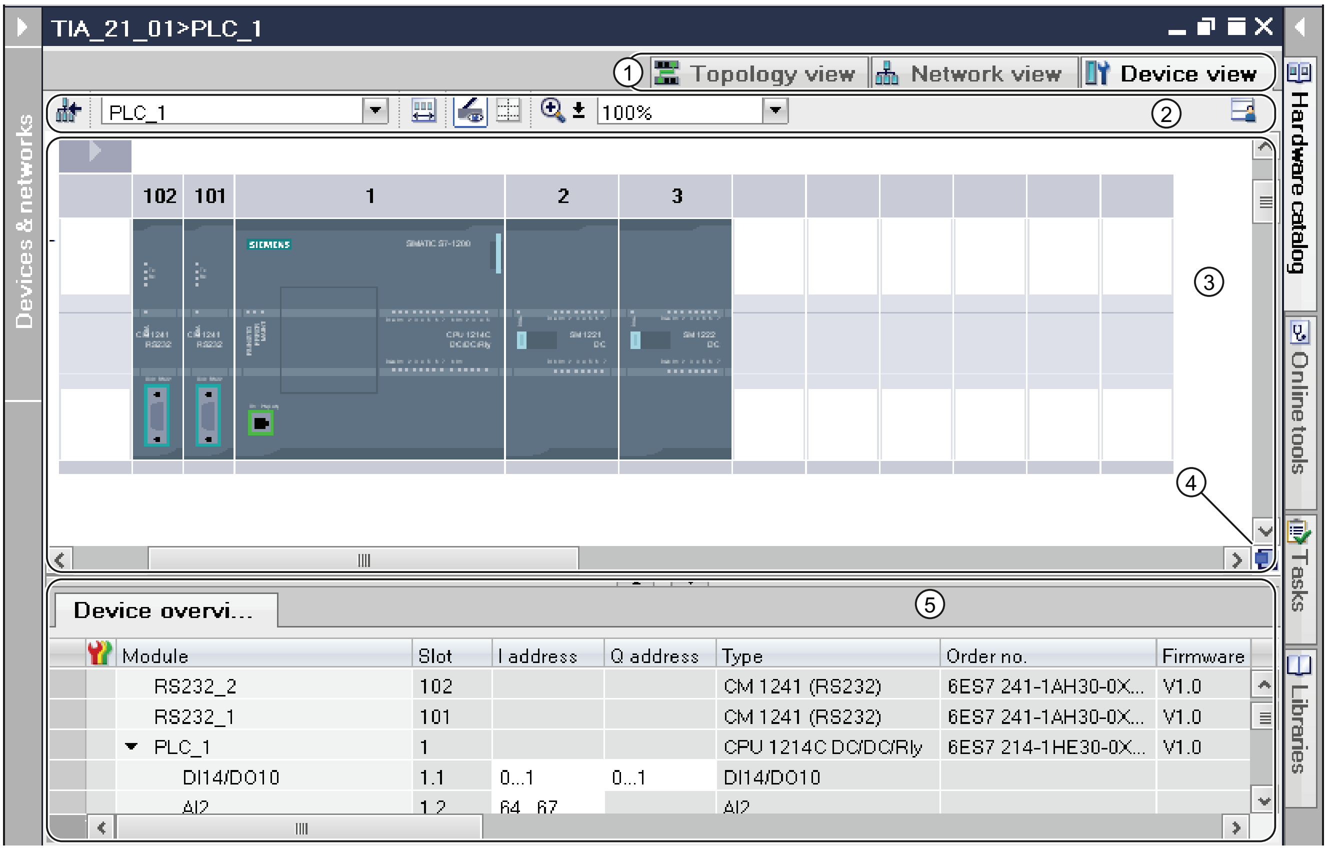
Introduction

The device view is one of three working areas of the hardware and network editor. You undertake the following tasks here:

- Configuring and assign device parameters
- Configuring and assign module parameters

Structure

The following diagram shows the components of the device view:



	Module	Slot	I address	Q address	Type	Order no.	Firmware
	RS232_2	102			CM 1241 (RS232)	6ES7 241-1AH30-0X...	V1.0
	RS232_1	101			CM 1241 (RS232)	6ES7 241-1AH30-0X...	V1.0
	▼ PLC_1	1			CPU 1214C DC/DC/Rly	6ES7 214-1HE30-0X...	V1.0
	DI14/DO10	1.1	0...1	0...1	DI14/DO10		
	AI2	1.2	64...67		AI2		

① Changeover switch: device view / network view / topology view

② Toolbar of device view

③ Graphic area of the device view

④ Overview navigation

⑤ Table area of device view

You can use your mouse to change the spacing between the graphic and table areas of the device view. To do this, click on the upper edge of the table view and expand or contract this by moving the mouse with the mouse button held down. You can use the two small arrow keys to minimize, maximize or select the latest table structure of the table view with just one click.

Toolbar

The toolbar provides the following functions:

Symbol	Meaning
	Switches to the network view. Note: The device view can switch between the existing devices using the drop-down list.
	Show the area of unplugged modules.
	Show module labels.
	Adjust the zoom setting. Select the zoom setting or enter it directly in the drop-down list. You can use the Zoom symbol to zoom in or out incrementally or to drag a frame around an area to be enlarged. With signal modules, you can recognize the address labels from a zoom level of 200% or higher.
	Show page breaks Enables page break preview. Dotted lines will be displayed at the positions where the pages will break when printed.
	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the device view displays hardware components and if necessary the associated modules that are assigned to each other via one or more racks. In the case of devices with racks, you have the option of installing additional hardware objects from the hardware catalog into the slots on the racks.

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate to the desired objects and display them in the graphic area.

Table area

The table area of the device view gives you an overview of the modules used and the most important technical and organizational data.

You can use the shortcut menu of the title bar of the table to adjust the tabular display.

See also

Working with racks (Page 329)

Network view (Page 307)

Area for unplugged modules (Page 332)

Inserting a module into a rack (Page 339)

Objects in the device view (Page 331)

Determination of online status and display using symbols (Page 649)

8.1.1.4 Topology view

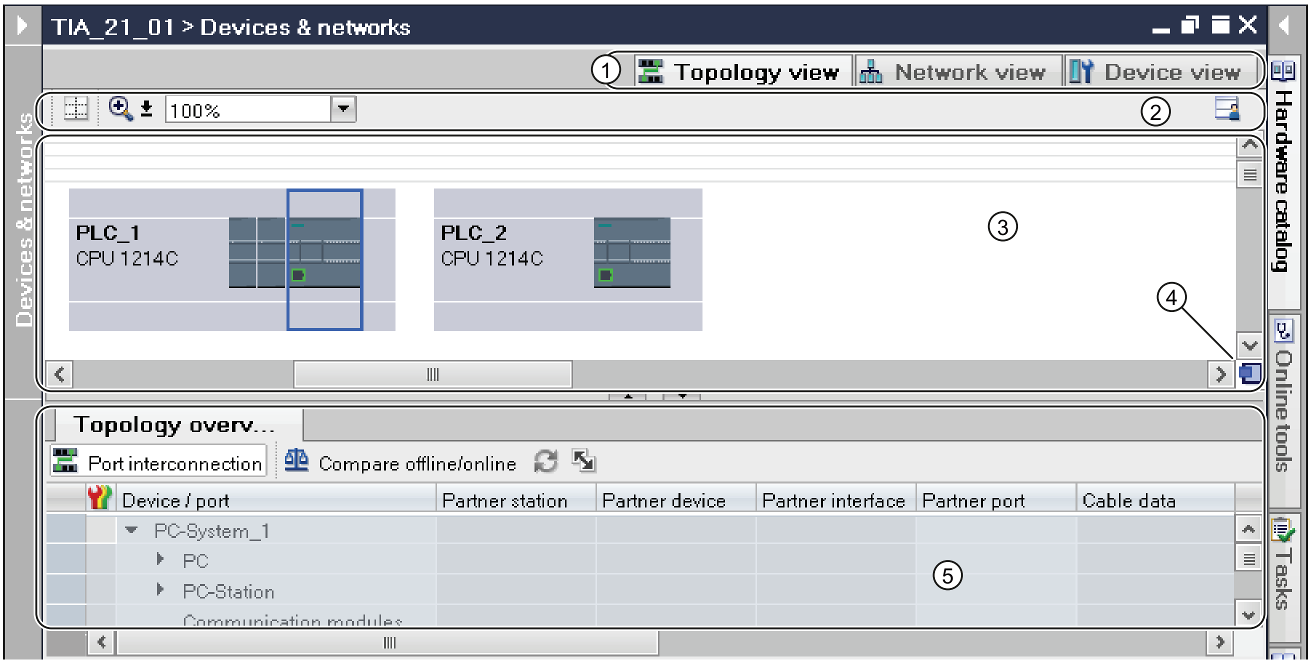
Introduction

The topology view is one of three working areas of the hardware and network editor. You undertake the following tasks here:

- Displaying the Ethernet topology
- Configuring the Ethernet topology
- Identify and minimize differences between the desired and actual topology

Structure

The following figure provides an overview of the topology view.



① Changeover switch: device view / network view / topology view
② Topology view toolbar
③ Graphic area of the topology view
④ Overview navigation
⑤ Table area of the topology view

You can use your mouse to change the spacing between the graphic and table areas of the topology view. To do this, click on the upper edge of the table view and expand or contract this by moving the mouse with the mouse button held down. You can use the two small arrow keys to minimize, maximize or select the latest table structure of the table view with just one click.

Toolbar

The toolbar provides the following functions:

Symbol	Meaning
🔍 ±	Adjusting the zoom setting. You can select the zoom setting via the drop-down list or enter it directly. You can also zoom in or zoom out the view in steps using the zoom symbol or draw a frame around an area to be zoomed in.
⊞	Show page breaks Enables page break preview. Dotted lines will be displayed at the positions where the pages will break when printed.
▤	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the topology view displays Ethernet modules with their appropriate ports and port connections. Here you can add additional hardware objects with Ethernet interfaces. See: Adding a device to the hardware configuration (Page 336)

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate to the desired objects and display them in the graphic area.

Table area

This displays the Ethernet or PROFINET modules with their appropriate ports and port connections in a table. This table corresponds to the network overview table in the network view.

You can use the shortcut menu of the title bar of the table to adjust the tabular display.

See also

Determination of online status and display using symbols (Page 649)

8.1.1.5 Printing hardware and network configurations

Printout of hardware and network configurations

You can print out the following elements of the hardware and network view as part of the project documentation:

- Graphic network view
- Network overview table
- Graphic device view
- The device overview table
- The parameters of the object currently selected in the editor

Printout of editor content

If you start a printout within an opened editor and no module is selected, the content of the editor is always printed. This includes a graphic representation of the editor as well as the table for the editor. You can adapt the scope of the printout. You can specify whether only the graphic view, the table or both together are to be printed. Read section "Changing the print options (Page 317)" for more on this.

If the graphic is larger than the page layout you have selected, the printout is continued on the next page. No content is lost this way. Alternatively, you can change the zoom level of the graphic representation to fit the printout on one page. The printout is always made in the currently selected zoom setting.

To check that all content fits on one page, you can either use the print preview or activate the page break preview. When page break preview is activated, dashed lines are displayed within the graphic editor at the location where the page break is later made.

Printing very large tables

If a table is larger than the print area and therefore cannot be fully printed, the content of the table is not printed as a table, but instead as pairs between value and key.

Example:

Object name	Property 1	Property 2
Object A	Value A1	Value A2
Object B	Value B1	Value B2

In this case, the printout has the following appearance:

Object A

Property 1: Value A1

Property 2: Value A2

Object B

Property 1: Value B1

Property 2: Value B2

You can also preset this as a template so that tables are always printed as pairs between the key and the value. Read section "Auto-Hotspot" for more on this.

Printing module parameters

Parameters of selected modules are printed out along with the current value settings in text form. All parameters from corresponding modules are also printed. For example, if you have selected a CPU, the parameters of an inserted signal board, if present, are printed as well.

You can determine the scope of the module parameters to be printed. In the "Print" dialog, select whether all properties and parameters of a module are to be printed or whether to use the compact printout. If you select the compact form, only the entries in the "General" area of the module properties are printed. Comments on modules, as well as the author and module description, are excluded. In compact mode, the following module parameters are therefore printed, for example:

- Module specifications

  Name, module slot, short description, order number, firmware version
- Name of the PROFINET interface
- Subnet specifications

  Name of the subnet, ID of the S7 subnet

See also

Changing the print options (Page 317)

Activating the page break preview for printout (Page 317)

8.1.1.6 Activating the page break preview for printout

You can activate the page break preview for the printout in the graphic editor. If this option is activated, dashed lines are shown within the graphic editor at the locations where page breaks are later made during printout.

Procedure

Proceed as follows to activate the page break preview:

1. Select the graphic area of the corresponding view.
2. Click on the "Show page break" symbol in the toolbar of the graphic editor.

   Dashed lines are displayed within the graphic editor at the location a page break is later made.
3. To modify the frame layout, select the "Print" command in the "Project" menu.
4. To disable page break preview, click again on the "Show page break" symbol in the toolbar of the graphic editor.

8.1.1.7 Changing the print options

Changing the scope of the printout

When printing from an editor, you can specify whether both graphics and tables are to be printed or just one of the two. Both are printed by default.

Procedure

To change the scope of the printout, proceed as follows:

1. In the "Options" menu, select the "Settings" command.
2. In the area navigation, open the "Print settings" parameter group under "General".
3. Scroll to the "Hardware configuration" group.
4. Select or clear the "Active graphic view" check box, depending on whether you want to print the graphics of the network and device view as well.
5. Select or clear the "Active table" check box, depending on whether you want to print the table for the editor as well.

See also

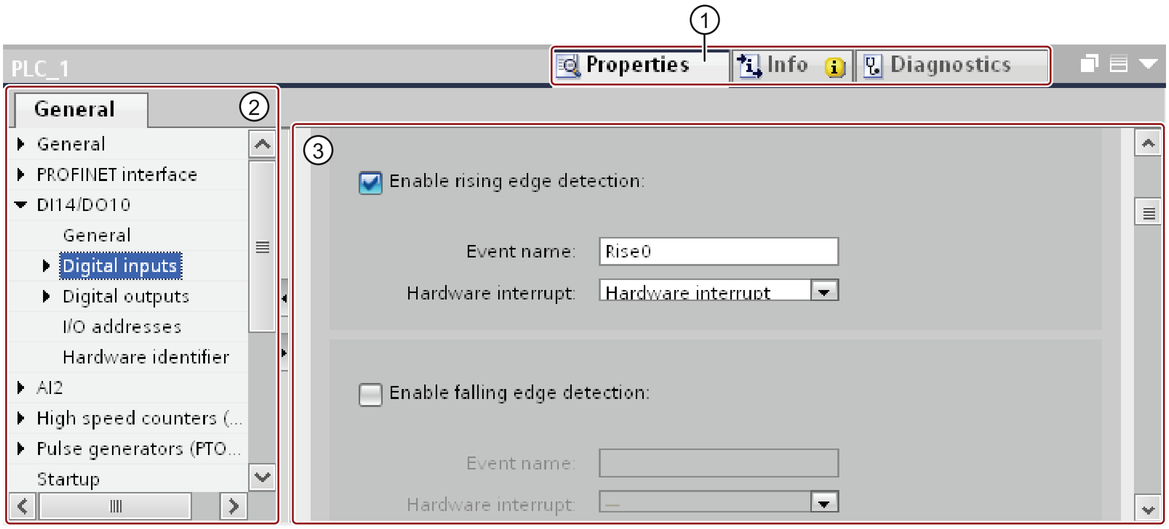
Printing hardware and network configurations (Page 315)

8.1.1.8 Inspector window

The properties and parameters shown for the object selected can be edited in the inspector window.

Structure

The inspector window consists of the following components:



☑ Enable rising edge detection:

Event name: Rise0

Hardware interrupt: Hardware interrupt

□ Enable falling edge detection:

Event name:

Hardware interrupt: ---

① Switch between various information and work areas

② Navigation between various pieces of information and parameters

③ Display showing the selected information and parameters

Function

The information and parameters in the inspector window are split into different types of information:

- Properties
- Info
- Diagnostics

To display the corresponding information and parameters, click in the area you want. The "Properties" area is the most important one for configuring an automation system. This area is displayed by default.

The left pane of the inspector window is used for area navigation. Information and parameters are arranged there in groups. If you click on the arrow symbol to the left of the group name, you can expand the group if sub-groups are available. If you select a group or sub-group, the corresponding information and parameters are displayed in the right pane of the inspector window and can be edited there too.

See also

Editing properties and parameters (Page 345)

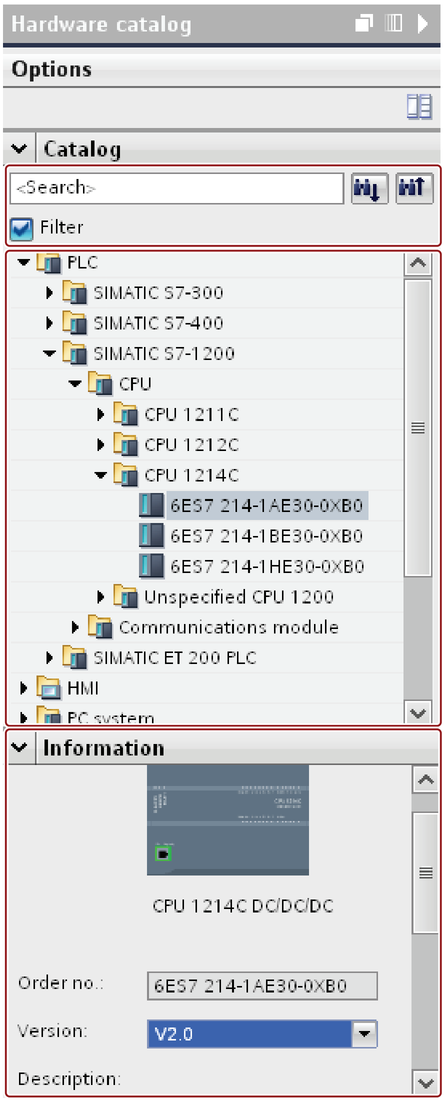
Overview of hardware and network editor (Page 305)

8.1.1.9 Hardware catalog

The "Hardware catalog" task card gives you easy access to a wide range of hardware components.

Structure

The "Hardware catalog" task card consists of the following panes:



Hardware catalog

Options

Catalog

<Search>

☑ Filter

- PLC
  - SIMATIC S7-300
  - SIMATIC S7-400
  - SIMATIC S7-1200
    - CPU
      - CPU 1211C
      - CPU 1212C
      - CPU 1214C
        - 6ES7 214-1AE30-0XB0
        - 6ES7 214-1BE30-0XB0
        - 6ES7 214-1HE30-0XB0
      - Unspecified CPU 1200
    - Communications module
  - SIMATIC ET 200 PLC
- HMI
- PC system

Information

CPU 1214C DC/DC/DC

Order no.: 6ES7 214-1AE30-0XB0

Version: V2.0

Description:

① "Catalog" pane, search and filter function
② "Catalog" pane, component selection
③ "Information" pane

Search and filter function

The search and filter functions of the "Catalog" pane make it easy to search for particular hardware components. You can limit the display of the hardware components to certain criteria using the filter function. For example, you can limit the display to objects that you can also place within the current context or which contain certain functions.

Objects that can be used in the current context include, for example, interconnectable objects in the network view or only modules compatible with the device in the device view.

Component selection

The component selection in the "Catalog" pane contains the installed hardware components in a tree structure. You can move the devices or modules you want from the catalog to the graphic work area of the device or network view.

Installed hardware components without a license are grayed out. You cannot use non-licensed hardware components.

Hardware components belonging to various components groups thematically are partially implemented as linked objects. When you click on such linked hardware components, a catalog tree opens in which you can find the appropriate hardware components.

Information

The "Information" pane contains detailed information on the object selected from the catalog:

- Schematic representation
- Name
- Version number
- Order number
- Brief description

See also

Browsing the hardware catalog (Page 327)

Overview of hardware and network editor (Page 305)

Information on hardware components (Page 322)

8.1.1.10 Information on hardware components

In the hardware catalog, you can display information on selected hardware components in the "Information" pane. You can also display further information on the selected hardware components using the shortcut menu.

Access to further information

If you select a hardware object in the hardware catalog and open the shortcut menu, you not only have the "Copy" function available but also three options for accessing information on Service & Support:

- Information regarding product support
- FAQs
- Manuals

The required information is displayed in the work area of the hardware and network editor.

Note

You can only access Service & Support when you are connected to the Internet and the function is enabled. By default, the function is disabled. To enable the function, refer to the instructions in the section "Enabling product support (Page 323)".

Information regarding product support

Here, you have access to general information on hardware and software components. The order number of the selected hardware object is entered automatically in the search mask. You can, however, also search for other hardware and software components.

FAQs

Here, you have access to "Frequently Asked Questions" (FAQs). You can view various entries on hardware and software questions. Using a detailed search mask, you can filter the required topics.

Manuals

Here, you have access to the manuals of the various hardware components. This is particularly useful if the configuration, addressing or parameter assignment you are planning requires more detailed knowledge of the hardware you are using.

See also

Hardware catalog (Page 320)

Enabling product support (Page 323)

8.1.1.11 Enabling product support

Enabling Service & Support function

For each device in the hardware catalog, you have the option of displaying additional information that is stored in the Service & Support area of the Siemens website. By default, the function is disabled. Instructions for enabling the function are given below.

Requirement

The software must have access to the Internet.

Procedure

To enable the Service & Support function, proceed as follows:

1. In the "Options" menu, select the "Settings" command.
2. Open the "Hardware configuration" group in the area navigation.
3. Select the "Via Internet" check box.

Result

You can now access product support, FAQs and manuals from the hardware catalog via the shortcut menu for the module.

See also

Information on hardware components (Page 322)

8.1.1.12 Keyboard action in the hardware and network editor

You can execute some of the functions of the network and device view directly with a combination of keyboard and mouse in the hardware and network editor. The keyboard operation in tables corresponds to standard behavior. Here you find the keyboard operation for the graphic work area of the network and device view.

General keyboard operation

Function	Key combination	Comments
Zoom in on view in frame	<Ctrl+Space> + mouse button pressed	No object selection
Move view	<Space> + mouse button pressed	-
Cancel current operation	<Esc>	-
Separate connection	<Esc> or double-click	When making a connection
Zoom in graphic view	<Ctrl> + turn mouse wheel	-

Selected objects

Function	Key combination	Comments
Select object	Mouse click	-
Move object	<Ctrl+X>, then <Ctrl+V>	Copy selected object to move and then paste in new position
Copy object	<Ctrl+C>	Copy selected object to clipboard
Paste object	<Ctrl+V>	Paste object from clipboard to selection
Delete selected object	<Del>	-
Select several objects 1	<Shift> + click	Select objects individually
Select several objects 2	<Ctrl> + click	Selected objects can also be deselected
Move selection	Mouse button pressed	Move to permitted slot
Copy selection	<Ctrl> + mouse button pressed	Move to permitted slot

8.1.2 Configuring devices

8.1.2.1 Basics

Introduction to configuring hardware

To set up an automation system, you will need to configure, assign parameters and interlink the individual hardware components. The work needed for this is undertaken in the device and network view.

Configuring

"Configuring" is understood to mean arranging, setting and networking devices and modules within the device or network view. Racks are represented symbolically. Just like "real" racks, they allow you to plug in a defined number of modules.

An address is automatically assigned to each module. The addresses can be subsequently modified.

When the automation system is started, the CPU compares the setpoint configuration produced by the software with the system's actual configuration. Possible errors can be detected and reported straight away.

Assigning parameters

"Assigning parameters" is understood to mean setting the properties of the components used. Hardware components and settings for the exchange of data are assigned:

- Properties of modules with selectable parameters
- Settings for data exchange between components

The parameters are loaded into the CPU and transferred to the corresponding modules when the CPU starts up. Modules can be replaced with ease since the parameters set are automatically loaded into the new module during startup.

Adjusting the hardware to the project requirements

You need to configure hardware if you want to set up, expand or change an automation project. To do this, add hardware components to your structure, link these with existing components, and adapt the hardware properties to the tasks.

The properties of the automation systems and modules are preset such that in many cases they do not have to be parameterized again. Parameter assignment is however needed in the following cases:

- You want to change the default parameter settings of a module.
- You want to use special functions.
- You want to configure communication connections.

See also

Changing properties of the modules (Page 546)

Using existing configurations

Open existing projects

When opening existing projects, an automatic check is made to determine if the appropriate software is installed for all modules used within the project. If you try to open a project with modules that are not supported by the current scope of the installation of the TIA portal, a message appears on opening the project informing you of the missing software components. If the software components are not absolutely required to open the project, the project can still be opened.

Reaction to missing software components

Projects that contain modules not supported by the current scope of the installation react as follows:

- Display the modules on the GUI
  - The non-supported modules are displayed in the project navigation with all of their nested objects. However, the modules themselves cannot be processed in editors or in inspector windows. When possible, a replacement module is used that best matches the original module. Replacement modules are indicated by an exclamation mark.
  - Display of properties in tables is limited. This applies in particular to the display of network parameters, such as the IP address.
- Functional limitations
  - Non-supported modules cannot be printed out or compiled.
  - An online connection cannot be established to the module. It is therefore also impossible to download.
  - To change the device type, the device must first be deleted and then re-inserted. The "Change device type" function is not supported.
  - Copying and inserting nested objects, such as modules, is possible although the device itself cannot be copied and inserted.
  - The network configuration cannot be changed with replacement modules within the network view.
  - Cross-references can be displayed. However, the cross-references only reflect the state last saved within the project because on online comparison to the original module cannot be made.

General slot rules

Introduction

Specific slot rules apply to each automation system and module.

If you select a module from the hardware catalog in the device view, all possible slots for the module selected are marked in the rack. You can only drag modules to marked slots.

If you insert, move or swap a module, the slot rules are also applied.

Consistency

Some slot rules depend on how the environment is configured. This means that you can sometimes plug modules into the rack although this would result in inconsistencies at the current time. If you change the configuration, for example by selecting different modules or module parameter settings, you can make the configuration consistent again.

In cases where inserting a module results in an inconsistency that can be corrected, this will be permitted. A consistency check is run when transferring the configuration. Inconsistencies are displayed as alarms in the inspector window under "Info". You can revise your configuration on the basis of the results of the consistency check and make it consistent again.

Rules for arranging modules

As a rule of thumb, the following applies to modules in racks:

- You can only plug modules into a rack.
- You can only plug interface modules into a module.
- You can only use modules of the same product or system family in one rack.

There are also other special rules for some modules:

- Can only be inserted in certain slots
- Inertion depends on other modules, CPUs or settings
- Limitation of the number of times used in a rack

Browsing the hardware catalog

Introduction

Use the "Hardware catalog" task card to select the hardware components you want for a hardware configuration. Use the hardware catalog to select the interconnectable hardware components in the network and topology view and to select the modules you want in the device view.

Context filter

You can use the "Filter" option of the hardware catalog to restrict the number of displayed hardware components and the number of hardware components that can be found by searching.

If you select the filter, only those components are displayed that can be selected currently in the hardware catalog. If the do not select the filter, the entire hardware catalog is displayed.

If you switch between the various views, the view of the filter objects is adapted to the current context.

Search options

You can use the search function to search for specific entries in the hardware catalog. Note the following rules when entering search terms:

- No distinction is made between upper and lower case text.
- Dashes and blanks are ignored during the search.
- The search function considers parts of a search term.
- Several search terms must be separated by a space

You start the search from an object highlighted in the hardware catalog and either search upwards or downwards.

Symbol	Meaning
↓	Downwards search
↑	Upwards search

Browsing the hardware catalog

If you want to browse the hardware catalog, proceed as follows:

1. Click in the entry field of the search function
2. Enter a search term. The search includes the following elements:
   - Name of device or module
   - Order number (MLFB)
   - Description in "Information" pane
3. Click on either the "Downwards search" or "Upwards search" buttons.

Note

To ensure the right search direction, note which point you have marked in the hardware catalog. To browse the entire catalog, click on the topmost object of the hardware catalog and start the search once you have entered the search term by clicking "Downwards search".

The first match with the search term found is displayed as the result. For more search results, again click on the "Downwards search" or "Upwards search" button.

Observe the context filter of the hardware catalog. If this is selected, the search in the hardware catalog is restricted to the displayed inserted hardware components.

See also

Hardware catalog (Page 320)

Information on hardware components (Page 322)

Working with racks

Introduction

To assign modules to a device, you need a rack, for example a mounting rail. Secure the modules on the rack and connect these via the backplane bus with the CPU, a power supply or other modules.

Creating a rack

If you insert a device in the network view, a station and a rack suitable for the device selected are created automatically. The rack and slots available are displayed in the device view. The number of slots available again depends on the type of device used.

Rack structure

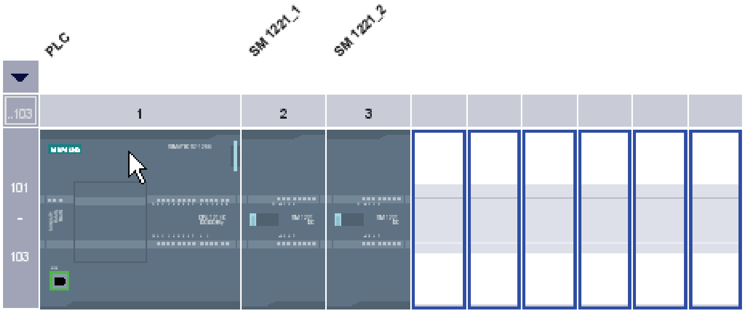
A rack always contains the device that has been inserted in the network view. The device is permanently assigned a slot which will depend on the type of device in question. There are additional slots on the right of the device and, if necessary, on left of the device; slot numbers are located above slots in which devices are plugged.

A corresponding short description is displayed above the plugged devices and modules. You show or hide this short description via the toolbar under "View" with the command "Display module titles" or the corresponding symbol in the toolbar of the device view (Page 310).

Symbol	Meaning
	Show module titles

When modules are selected in the hardware catalog, all the slots permitted for this module are marked. This allows you to see immediately the slot into which the selected module can be inserted.

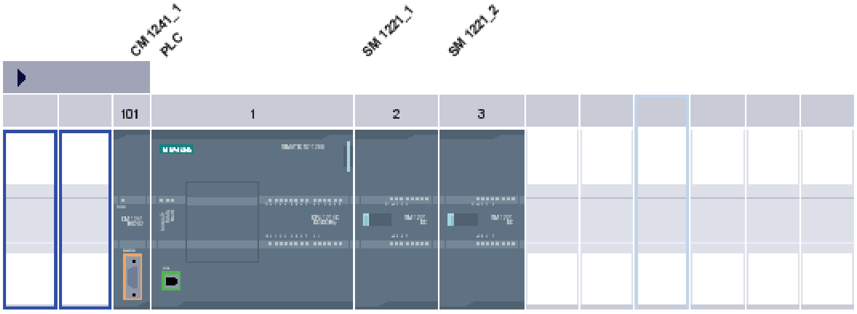
In the following screenshot, a signal module has been selected in the hardware catalog for a partially filled S7-1200 rack:



Since slots 101-103 are reserved for communications modules, only the other free slots are shown as available slots.

You can expand and collapse the front group of slots using an arrow symbol above the expandable slot. When the group of slots is collapsed, the first and last of the group's slot numbers are displayed.

The following figure shows the expanded slot group:



Groups of slots into which modules have already been plugged cannot be collapsed.

Multiple selection of modules and slots

There are various ways of selecting several modules or slots:

- By pressing <Shift> or <Ctrl>, you can select several modules or slots at the same time.
- Click outside the rack and then hold the mouse button and drag a frame to include the modules or slots you want to select.

Objects in the device view

A graphic display of the rack and the devices plugged into it is shown in the upper section of the device view.

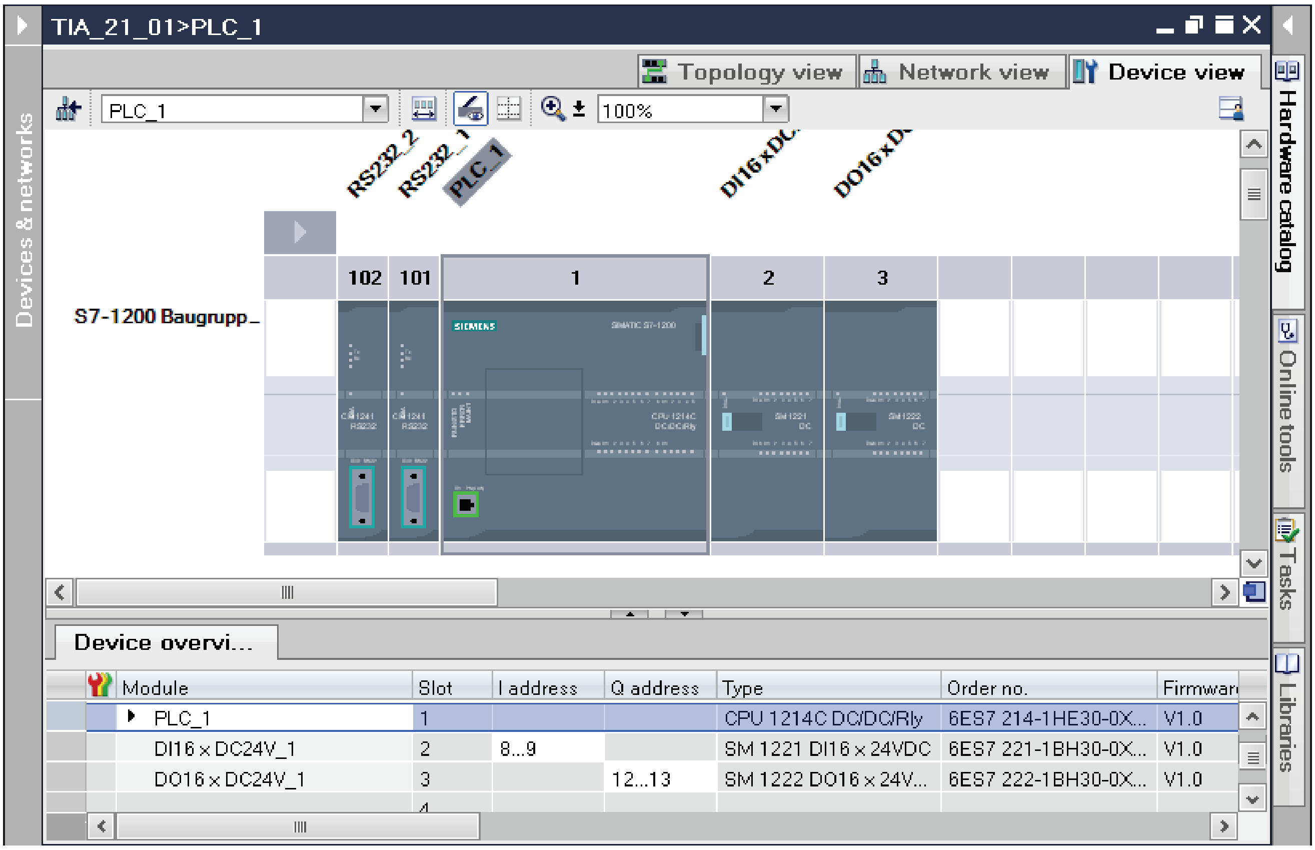
You can see the device overview in the bottom part of the device view. The device overview is a table showing you the most important information about the modules inserted in the rack.

Structure and content of device view

The offline configuration of the devices in the rack are displayed in the graphic device view. This is a symbolic representation of the configuration on the real rack.

The rack configuration is displayed as a table in the device view. Each line in the table contains the information for assigning a slot.

The following screenshot shows the device view with the configuration of a SIMATIC S7-1200 CPU.



Module	Slot	I address	Q address	Type	Order no.	Firmwar
PLC_1	1			CPU 1214C DC/DC/Rly	6ES7 214-1HE30-0X...	V1.0
DI16 x DC24V_1	2	8...9		SM 1221 DI16 x 24VDC	6ES7 221-1BH30-0X...	V1.0
DO16 x DC24V_1	3		12...13	SM 1222 DO16 x 24V...	6ES7 222-1BH30-0X...	V1.0
	4					

In the upper part, you can see the graphic view showing how the rack is occupied by various modules in slots 1 to 3 as well as 101. In the lower part you can see a tabular representation of the rack in the device overview.

Each line in the device overview represents one slot. The key information for each slot is displayed in the various columns:

Column	Meaning
Module	Name of module, can be edited in any way
Slot	Slot number
I address	Input address area, can be edited in any way
Q address	Output address area, can be edited in any way
Type	Catalog name of module
Order no.	Module order number
Firmware	Firmware version of module
Comments	Optional comments

See also

Device view (Page 310)

Area for unplugged modules

In some cases, the modules for a hardware configuration are not assigned a slot for short periods. Such unplugged modules are moved to the area of unplugged modules, a special area in the device view.

Adding modules to the storage area

The modules, which for example are to be assigned to a device using a copy action but for which the corresponding rack does not have a free compatible slot, are moved automatically into the area of unplugged modules.

Under the following conditions, modules are automatically added to the area of unplugged modules:

- In the network view, a module is moved to a device but the rack does not have a compatible free slot.
- In the device view, a module is moved from the rack, the hardware catalog or the project tree straight into the storage area.

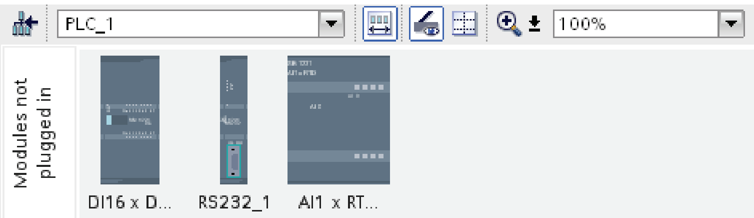
CPs and FMs which occupy a network resource can be moved into the area for unplugged modules but will lose the network resources they have been assigned.

You can add modules to the area of unplugged modules by means of drag-and-drop, for example. To do this, the area must be opened.

Using the area of unplugged modules

Use the corresponding button to open the area of unplugged modules.

You can find the area of unplugged modules in the device view.



You open the area of unplugged modules with the respective symbol in the toolbar of the device view (Page 310).

Symbol	Meaning
▤	Open area of unplugged modules

Note

To free up slots, move modules from your configuration into the storage area and plug the modules you want from the storage area into the freed up slots.

You can use this approach to temporarily move modules that have already been parameterized out of the configuration without deleting them.

Treatment of modules in the storage area

The following rules apply to modules in the storage area:

- The modules appear in the project tree under the corresponding device in the "Local modules" folder.
- The modules retain all settings and parameters previously provided.
- The modules are not taken into account when downloading to a target system so a consistency check is not undertaken for modules in the area of unplugged modules.
- Using the context menu, the modules can be copied, pasted or deleted, for example.

8.1.2.2 Configuring individual devices

Selecting a CPU

Introduction

Select a CPU from the hardware catalog and place it, together with a rack, in the network view. On this device drag the desired modules from the hardware catalog; they are arranged automatically on the rack.

Selecting the components in the hardware catalog

Each hardware component is displayed as a folder in the hardware catalog. When you open this folder you will see the different versions of the selected hardware component with its respective order numbers.

There will be an example of how to set up a CPU with a rack in network view.

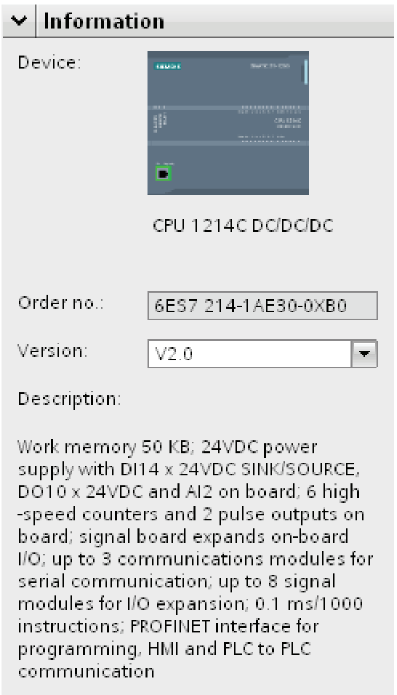
Requirement

- The hardware catalog is open.
- You must be in the network view.

Procedure

To select a CPU from the hardware catalog, proceed as follows:

1. In the hardware catalog navigate to the folder with the desired CPUs.
2. Open the folder with the desired CPU type; you will see all order numbers for the selected CPU type.
3. Click on a CPU order number to get information about the selected CPU under "Information" pane.



Information

Device:

CPU 1214C DC/DC/DC

Order no.: 6ES7 214-1AE30-0XB0

Version: V2.0

Description:

Work memory 50 KB; 24VDC power supply with DI14 x 24VDC SINK/SOURCE, DO10 x 24VDC and AI2 on board; 6 high -speed counters and 2 pulse outputs on board; signal board expands on-board I/O; up to 3 communications modules for serial communication; up to 8 signal modules for I/O expansion; 0.1 ms/1000 instructions; PROFINET interface for programming, HMI and PLC to PLC communication

4. Set up the CPU and a rack. You have the following options:
   - Use drag-and-drop to drag the CPU from the hardware catalog into network view.
   - Use Copy & Paste to copy the CPU to the network view.
   - Double-click the CPU entry in the hardware catalog.

See also

Browsing the hardware catalog (Page 327)

Adding a device to the hardware configuration (Page 336)

Inserting a module into a rack (Page 339)

Working with racks (Page 329)

Creating an unspecified CPU (Page 337)

Information on hardware components (Page 322)

Adding a device to the hardware configuration

Introduction

There are various ways of adding a connectable device from the hardware configuration in the network view and the topology view:

- Command "Add new device" in the project tree
- Double-click device in hardware catalog
- Drag-and-drop from the hardware catalog in network view or in topology view:
  - Text entry from the "Catalog" pane
  - Preview graphic from the "Information" pane
- "Add > Device" command from menu bar in network view or topology view
- Shortcut menu of a device in the hardware catalog for copying and pasting

A suitable rack is created along with the new device. The selected device is inserted at the first permitted slot of the rack.

Regardless of the method selected, the added device is visible in the project tree and the network view of the hardware and network editor.

Adding device using the project tree

To use the project tree to add a device to the hardware configuration, proceed as follows:

1. Click on the command "Add new device" in the project tree.

   The "Add new device" dialog box opens.

2. Display the required device in the tree structure:
   - Go to required device in the tree structure.
   - Enter a device name in the entry field.
3. Select the required device from the tree.

   More information about the device presently selected is displayed on the right-side of the dialog box.

4. If necessary, set the firmware version using the drop-down list in the dialog box.
5. Select the "Open device view" check box if you want to change to the device view immediately after adding the device.

   There you can immediately continue with device configuration and equipping the rack.

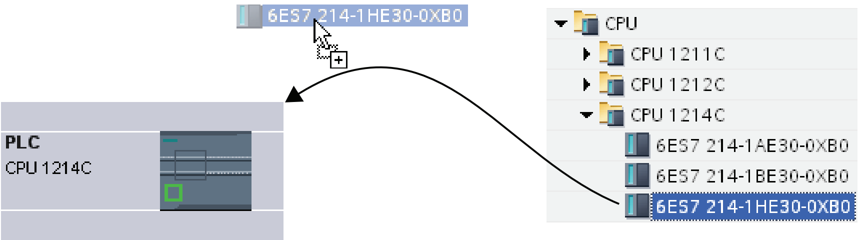
6. Click on "OK" to add the device selected.

   The dialog box closes.

Adding device from the hardware catalog

To add a device to the hardware configuration using the hardware catalog, proceed as follows:

1. Open the network view or the topology view.
2. Open the hardware catalog.
3. Go to the required device in the hardware catalog.
4. Click on the chosen device to select it.
5. If necessary, set the firmware version using the drop-down list in the hardware catalog.
6. Drag the device to the network view or the topology view.



You have now placed the device in the network view or in the topology view. The displayed rectangle (in other words "Station") symbolizes the plugged device together with its rack and any lower-level modules. Double-click on the device or station to open the device view and view the new rack and inserted device. In the next steps, you can configure the device in the device view and equip the rack with modules.

See also

Network view (Page 307)

Creating an unspecified CPU (Page 337)

Information on hardware components (Page 322)

Topology view (Page 313)

Creating an unspecified CPU

Introduction

If you have not yet selected a CPU but have already started programming or want to use an existing program, you have the option of using an unspecified CPU. You can also adjust some settings with unspecified CPUs. The setting options are restricted to parameters that all CPUs of the same CPU family have in common.

Creating an unspecified CPU in the portal view

To create an unspecified CPU in the portal view, follow these steps:

1. Now, click one of the following options:
   – "Devices & networks > Add new device"
   – "PLC programming" > "Device" button
2. For a device family, select an unspecified CPU from the tree structure of the "Add new device" dialog.
3. Click on "Add".

An unspecified CPU is created and the device view for this CPU appears.

Further options for creating unspecified CPUs

In the project view, you can create unspecified CPUs like specified CPUs:

- Using the "Add new device" button in the project tree
- In the "Hardware catalog" task card

You can also use these methods to create multiple unspecified CPUs.

Specifying unspecified CPUs

You have two options for specifying unspecified CPUs:

- Use drag-and-drop to assign an existing CPU from the hardware catalog to an unspecified CPU by means of module replacement (Page 344).
- Select an unspecified CPU and then click "Online > Hardware detection" in the menu bar and assign a CPU identified online. For this purpose, you assign an IP address using the "Add address for PG/PC" button.

See also

Selecting a CPU (Page 334)

Adding a device to the hardware configuration (Page 336)

Inserting a module into a rack

Introduction

Once you have added devices from the hardware catalog to your configuration in network view, you can add modules to the devices. There are various ways of adding a module to a rack in the device view:

- If there is an available valid slot, double-click a module in the hardware catalog.
- Use drag-and-drop to move the module from the hardware catalog to an available valid slot in the graphic or table area:
  - Text entry from the "Catalog" pane
  - Preview graphic from the "Information" pane
- Select "Copy" in the shortcut menu for a module in the hardware catalog and then select "Paste" in the shortcut menu on an available valid slot in the graphic or table area.

To access the device view from the network view, double-click a device or station in the network view or select the Device view tab. The device view contains an illustration of the device selected within a rack. The graphic illustration of the rack in the software corresponds to the real structure, i.e. you can see the same number of slots as exist in the real structure.

Note

You can also move a module to a rack in the network view. The filter function for the hardware catalog must be deactivated in this instance. The module is automatically plugged into a free and permitted slot. If there are no slots available, the module will be moved to the area of unplugged modules (Page 332).

Equipping a rack

Arrange the modules on a rack according to the applicable slot rules.

After a module has been inserted in a rack with an already inserted CPU, the address areas are checked automatically so that addresses are not assigned twice. After it has been inserted, each module then has one valid address range. To do so, DP slaves and IO devices must be networked with a CPU via the corresponding DP master or IO system.

Requirements

- You are in the device view.
- The hardware catalog is open.

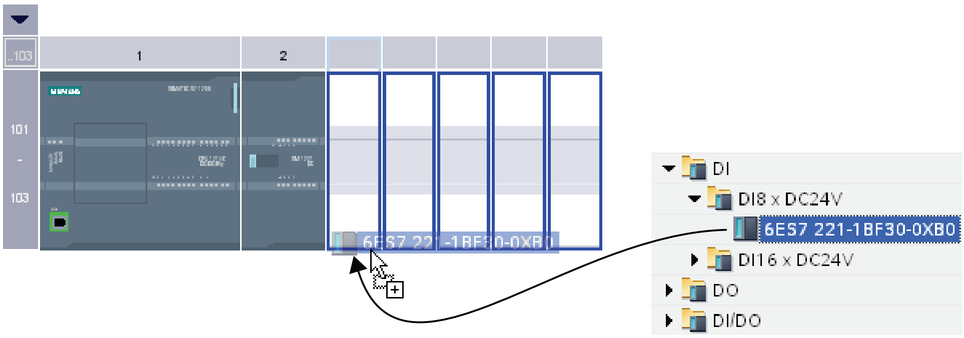
Adding module from the hardware catalog

How to insert a module from the hardware catalog into a rack is illustrated based on the example of a signal module. To do this, follow these steps:

1. Go to the required module board in the hardware catalog.

> **Note**
>
> If you activate the filter function of the hardware catalog, only those modules which match the selected device type will be displayed.

2. Select the chosen module.
3. If necessary, set the firmware version using the drop-down list in the hardware catalog.
4. Drag the signal module to a free slot in the rack.

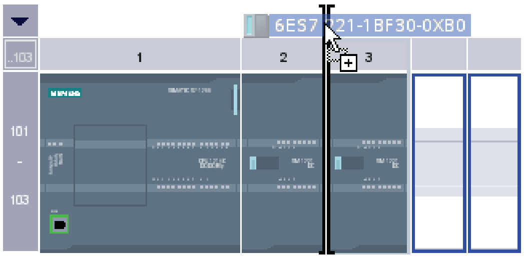


You have now inserted the digital signal module in a slot in the rack. Repeat these steps with the other modules.

The name of the module is displayed above the inserted modules. You can activate or deactivate module labeling in the menu bar with "View > Show module labels".

Inserting module

You can also drag modules and drop them between modules that have already been inserted. To do this, drag a module above and between the two existing modules while holding down the mouse button.



A mouse pointer appears. When you release the mouse button, all modules plugged to the right of the pointer are moved one slot to the right. Any redundant modules are moved to the area of unplugged modules. The new module is plugged at the point of the freed up slot.

See also

Device view (Page 310)

Area for unplugged modules (Page 332)

Information on hardware components (Page 322)

General slot rules (Page 327)

Deleting a hardware component

There are various ways of deleting hardware components. Deleted hardware components are removed from the system and assigned addresses made available again.

Rules

- CPUs or modules from the rack and from the area of unplugged modules can be deleted.
- When a rack is deleted in the device view the plugged hardware components are moved to the area of unplugged modules.

Procedure

Proceed as follows to delete a hardware component:

1. Select the hardware components you want to delete.
   - Network view: Select devices or network relevant hardware components in the graphic view or in the network view.
   - Device view: In the graphic view or device overview, select racks or modules in racks or in the area of unplugged components.
   - Topology view: Select devices or hardware components with Ethernet interfaces in the graphic view or in the topology view.
   - Project tree: Select devices or individual hardware components from the tree structure.
2. Select "Delete" from the shortcut menu or press <Del>.

   If the "Delete" menu item is unavailable, your selection contains at least one component that cannot be deleted.

The selected hardware components are deleted.

Note

Deleting hardware components may result in inconsistencies in the project, for example infringement of slot rules. Inconsistencies are reported during the consistency check. Correct the inconsistencies by taking appropriate action, for example, make sure that slot rules are kept to.

See also

Keyboard action in the hardware and network editor (Page 324)

Copying a hardware component

You can copy hardware components in the device or network view. Copied hardware components are stored on a clipboard and can be pasted at another point from this clipboard. Copied stations are pasted as new stations in the network view. Copied devices and modules can be pasted into existing racks in the network and device view.

Rules

- Single objects as well as several objects can be copied at the same time.
- Modules inserted in the rack and in the area of unplugged modules can be copied.
- You can only copy devices and modules to free and valid slots in keeping with the slot rules.
- Racks with a CPU inserted cannot be copied individually, but only as complete units along with all inserted hardware components.

Procedure

Proceed as follows to copy a hardware component:

1. Select the hardware components you want to copy.
   - Device view: Select the module in a rack or put it in the area of unplugged modules.
   - Network view: Select the station or the relevant hardware component from the network view.
   - Project tree: Select the station or module.
2. Select "Copy" from the shortcut menu or press <Ctrl+C>.

   If the "Copy" menu item is unavailable, your selection contains at least one component that cannot be copied.
3. Select the location at which the content of the clipboard is to be pasted.
   - Device view: Select a free slot in the rack or area of unplugged modules.
   - Network view: Select a station where you want to insert devices or modules or move the mouse pointer to a free location in the network view to paste a copied station or a hardware component relevant to the network view.
4. Select "Paste" from the shortcut menu or press <Ctrl+V>.

   If the "Paste" menu item is unavailable, the clipboard is empty or contains at least one component that cannot be pasted at this point.

The selected object is pasted at the chosen point.

Once you have selected a station where you want to insert a module in the network view, the module is inserted in the first free and valid slot. If no free, valid slots are available, the object is inserted in the area of unplugged modules.

Note

You can also copy a module from one device to another:

To do so, copy a module in the hardware and network editor, select a different device in the network view or the drop down list of the device view and insert the module.

You can insert the copied object directly in a slot or place it in the area of unplugged modules in the device view. If you add the copied object in the network view of a device or a station, it will be inserted in the first available slot.

If there is no slot available for the object, it is automatically placed in the area of unplugged modules (Page 332).

Note

You can use <Ctrl> and drag-and-drop to directly copy a selected hardware component.

See also

Keyboard action in the hardware and network editor (Page 324)

Moving a hardware component

You can move hardware components in the device or network view.

Rules

- You can move devices and modules from the rack and the area for unplugged modules taking the slot rules into consideration.
- CPs can be moved in the network view. The CP is plugged in a free and valid slot in the target device. If there are no free slots available, the CP to be inserted is moved to the area for unplugged modules.
- In the network view, CPU and slave head modules can be moved between the devices; depending on CPU type also within the rack.

Note

Moved CPs are disconnected from their network but keep their network parameters and address. If you reconnect the CP to the network and its address has been assigned, use a dialog to assign a new unique address to the CP.

Procedure

Proceed as follows to move a hardware component:

1. Select the hardware component you want to move.
   – Device view: Select the module in a rack or put it in the area of unplugged modules.
   – Network view: Select the hardware component of relevance to the network view.
2. Select "Cut" from the shortcut menu or press <Ctrl+X>.

   If the "Cut" menu item is unavailable, your selection contains at least one component that cannot be cut.
3. Select the location to which the cut object is to be moved.
   – Device view: Select a free slot in the rack or area of unplugged modules.
   – Network view: Select a station where you want to insert devices or modules.
4. Select "Paste" from the shortcut menu or press <Ctrl+V>.

   If the "Paste" menu item is unavailable, the clipboard is empty or contains at least one component that cannot be pasted at this point.

The selected hardware component is moved to the target. If the hardware component being moved is a networked object, it is disconnected from the network.

Note

You can use drag-and-drop to directly move a selected hardware component.

See also

Keyboard action in the hardware and network editor (Page 324)

Replacing a hardware component

You can replace hardware components with others. This, for example, allows you to replace unspecified CPUs (Page 337) with available CPUs from the hardware catalog.

Rules

You can only replace hardware components if they support module replacement and if the two components are compatible.

Procedure

To replace one module with another, proceed as follows:

1. Select the module you want to replace.
2. Open the shortcut menu:
   - If the "Replace device" entry is enabled, the module can be replaced.
   - If the "Replace device" entry is disabled, a module cannot be replaced.
3. Click on "Replace device" in the shortcut menu. ### The "Replace device" dialog box appears.
4. Under "New device" in the tree structure, select the module with which you want to replace your current module.
5. Click "OK".

The existing module is replaced by the new one.

As an alternative, you can take a module by dragging it from the hardware catalog to the module you are replacing. If the module can be replaced by the selected module, this is indicated by the mouse pointer symbol.

Editing properties and parameters

Once you have inserted hardware components in your rack, you can edit their default properties, for example parameters or addresses in the network or device view.

Requirements

You are in the device view.

Note

You can also edit properties and parameters in the network view. In the graphic network view, you have access to the network-related hardware components and the station. You can access modules and hardware components not displayed in the graphic network view using the table network view.

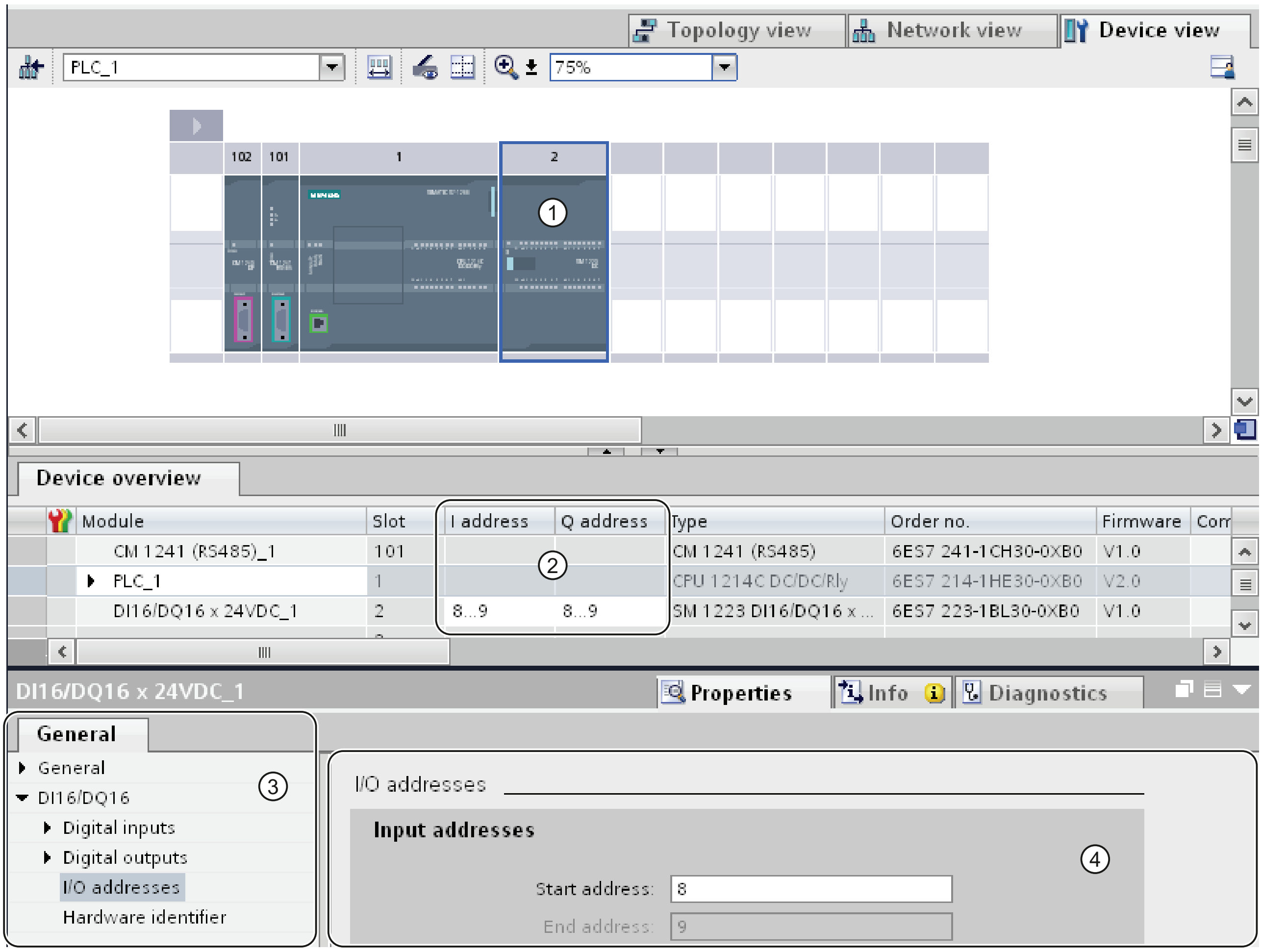
Procedure

To change the properties and parameters of the hardware components, proceed as follows:

1. In the graphic view, select the CPU, module, rack or interface you want to edit.
2. Edit the settings for the selected object:
   - Use the table view to edit addresses and names, for example.
   - In the Inspector window additional setting possibilities are available in "Properties".

Note that modules can only be fully parameterized if they are assigned to a CPU. Therefore, PROFIBUS or PROFINET interfaces modules must first be networked with the CPU or a centrally inserted communication module so that they form a master system or IO system. Only then is it possible, for example, to edit the addresses of the distributed components that are inserted.

Example of changing settings



Module	Slot	I address	Q address	Type	Order no.	Firmware	Com
CM 1241 (RS485)_1	101			CM 1241 (RS485)	6ES7 241-1CH30-0XB0	V1.0	
PLC_1	1			CPU 1214C DC/DC/Rly	6ES7 214-1HE30-0XB0	V2.0	
DI16/DQ16 x 24VDC_1	2	8...9	8...9	SM 1223 DI16/DQ16 x ...	6ES7 223-1BL30-0XB0	V1.0	

Input addresses

Start address: 8

End address: 9

① Selection of a module
② Editing option for addresses in the device overview
③ Selection options in the inspector window
④ Editing option for addresses in the inspector window

See also

Inspector window (Page 318)

Update module version

Explanation of terms

The terms "Module version" and "Firmware version" are explained in more detail in the following section.

- Module version: The specific version of the configuration software from which the module description stems.

  Exp.: V11.0.0.0

- Firmware version: The version of the firmware of the offline configured module

  Exp.: V2.0

Requirements

- You have created a device configuration.
- You have installed an update or an optional package at a later date. As a result of this installation, the module version of at least one module type was updated in the hardware catalog, whereby the new version is incompatible with the previous version.
- You have used such modules in your device configuration and want to use the modified or added properties.

Procedure

Perform the following step for each affected module type.

1. Select the affected module in the device view.
2. In the Inspector window, go to "Properties > General > Catalog Information". Click the "Update module version" button there.
3. In the query that then appears, specify whether you want to update the module version only for the selected module or for all modules of this type in the current project.

Result

The selected modules are replaced by the same modules with updated module version in the the current project.

In which cases is it unnecessary to update the module version?

An updating of the module version is not necessary in the following cases:

- You do not want to used the modified or added properties of the modules.
- You open an existing project with a version of the configuration software that is more recent than the version with which you created the project and the system automatically performs a project conversion, for example from V10.5 to V11.0. In this case, all older module versions are automatically adapted.

8.1.3 Configure networks

8.1.3.1 Networking devices

Communication and networks

Communication between devices

The basis of all types of communication is always a previously configured network. The network configuration provides the necessary requirements for communication:

- All the devices in a network are provided with a unique address
- Communication of the devices with consistent transmission properties

Network configuration

The following steps are necessary when configuring networks:

- Connect devices to subnet
- Specify the properties/parameters for each subnet
- Specify the device properties for every networked module
- Download configuration data to the devices to supply interfaces with the settings resulting from the network configuration
- Document the network configuration

For Open User Communication, creating and configuring a subnet is supported by the assignment of connection parameters.

Relation between network configuration and project

Within a project, subnets and their properties are managed. Properties result mainly from adjustable network parameters and the quantity and communication properties of the connected devices.

The devices to be networked must be in the same project.

Subnet name and subnet ID

Within the project, subnets are clearly identified by a subnet name and ID. The subnet ID is saved in all components along with interconnectable interfaces. Components can then be clearly assigned to a subnet even after uploading into a project.

Networking options

In the project, you can create and network devices with components capable of communication. The following basic options are available for networking the devices:

- You link the interfaces of the components capable of communication with one another. A new subnet is created suitable for the type of interface.
- You connect the interface of the devices capable of communication with a new or existing subnet.
- You create an Open User Communication connection. When you assign parameters to the connection for Open User Communication, a subnet is created automatically between the communication partners.
- You use the graphic connection configuration to configure connections; missing networks are hereby recognized and are created either automatically or via dialog.

Due to the different tasks of the devices or the span of the plant, you may need to use several subnets. These subnets are managed in a project.

Networking devices in the network view

Options

In the graphic network view, you have an overview of the subnets of the entire system in the project. You can use the tabular network overview for additional support.

Depending on the starting situation, there are various ways of undertaking configuration to network the interface for a component capable of communication. The procedures are described in the following section:

- Creating an individual subnet
- Creating several subnets at one time
- Connecting two target devices via a new subnet
- Connecting devices to existing subnet
- Selecting an existing subnet from a list
- Automatic networking during the configuration of the connection;

  See also: Auto-Hotspot

Possible starting situations are:

- A suitable subnet is not yet available.
- The subnet with which you want to connect the component already exists.

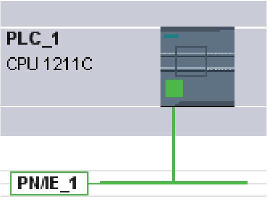
Procedure - creating a single subnet

To create a subnet and to connect it to an interface, proceed as follows:

1. Select the interface of a CPU / a CP.
2. Select the "Create subnet" command in the shortcut menu of the interface.

The selected interface is connected to a new subnet. Consistent address parameters are set automatically for the interface.

The following schematic shows an interface with outgoing line connecting to a subnet:



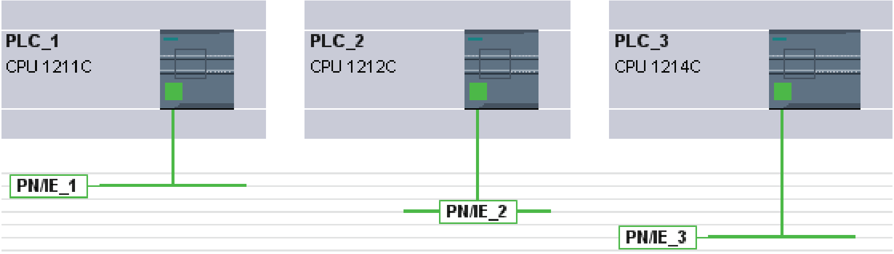
Procedure - creating several subnets at one time

To create several subnets at one time, proceed as follows:

1. Select several interfaces by clicking on them while pressing the <Ctrl> button.
2. Select the "Create subnet" command in the shortcut menu of the interface.

Each selected interface is connected to a new subnet. Consistent address parameters are set automatically for the interface.

The following figure shows multiple subnets created by selecting multiple interfaces:

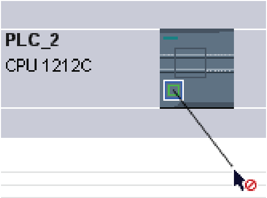


Procedure – Connecting two target devices via a new subnet

To connect an interface with another device via a subnet that does not yet exist, proceed as follows:

1. Position the mouse pointer over the interface for a component capable of communication requiring networking.
2. Click with the left mouse button and hold the button down.
3. Move the mouse pointer.

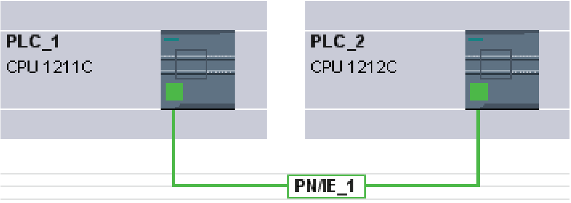
   The pointer now uses the networking symbol to indicate "Networking" mode. At the same time, the mouse pointer shows the lock symbol that will only disappear when the pointer is on a valid target.

4. Now move the pointer onto the interface of the target device. You can either keep the mouse button pressed or release it.
5. Now release the left mouse button or press it again (depending on previous action).

A new subnet is created. The interfaces are now connected via the new subnet. Consistent address parameters are set automatically for the interface.

The following schematic shows two networked devices:

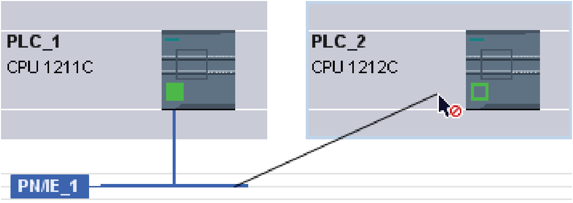


Procedure - Connecting devices to existing subnet

To connect an interface to an existing subnet, proceed as follows:

1. Position the mouse pointer on the interface of a communications-compliant component you want to network or on the existing subnet.
2. Click with the left mouse button and hold the button down.
3. Move the mouse pointer.

   The pointer now uses the networking symbol to indicate "Networking" mode. At the same time, the mouse pointer shows the lock symbol that will only disappear once the pointer is moved to a valid target.
4. Now move the mouse pointer to the existing subnet or to the interface to be networked. You can either keep the mouse button pressed or release it.



5. Now release the left mouse button or press it again (depending on previous action).

Result:

The interface and selected subnet are now connected. Consistent address parameters are set automatically for the interface.

Procedure - selecting an existing subnet from a list

To link an interface with a subnet that has already been created, proceed as follows:

1. Select the interface of a CPU.
2. Select the "Assign to new subnet" command in the shortcut menu of the interface.

   A list box containing the available subnets appears.
3. Select a subnet from the list.

The interface and selected subnet are now connected. Consistent address parameters are set automatically for the interface.

Tabular network overview

Meaning

The tabular network overview adds the following functions to the graphic network view:

- You obtain detailed information on the structure and parameter settings of the devices.
- Using the "Subnet" column, you can connect components capable of communication with created subnets.

Basic functions for tables

The network overview supports the following basic functions for editing a table:

- Displaying and hiding table columns

  Note: The columns of relevance to configuration cannot be hidden.
- Optimizing column width
- Sorting table
- Displaying the meaning of a column, a row or cell using tooltips.

Networking devices in the device view

Networking in the device view

In the device view, you can check and set all the parameters of the components belonging to a device and the interfaces in detail. Here you can also assign the interfaces to the subnets created in the project.

Requirements

- The subnet with which you want to connect an interface has already been created.
- If the subnet has not yet been created, change to the network view and make the settings required for networking.

Procedure - connecting to an existing subnet

To connect an interface to an existing subnet, proceed in the device view as follows:

1. Select the entire communications-compliant component or the interface to be networked. The properties of the selected interface or component are displayed in the Inspector window.
2. In the Inspector window, select the parameter group for the selected interface; for example, the "Ethernet addresses" parameter group for a PROFINET interface.
3. Select the subnet to be connected from the "Interface connected with" drop-down list.

The interface and selected subnet are now connected. Consistent address parameters are set automatically for the interface.

Procedure - creating a new subnet

To create a subnet and to connect it to the interface, proceed as follows in the device view:

1. Select the entire communications-compliant component or the interface to be networked. The properties of the selected interface or component are displayed in the Inspector window.
2. In the Inspector window, select the parameter group for the selected interface; for example, the "Ethernet addresses" parameter group for a PROFINET interface.
3. In "Interface connected with", click the "Add new subnet" button.

The interface is connected to a new subnet of the appropriate subnet type. Consistent address parameters are set automatically for the interface.

Checking or changing network parameters and interface parameters

Introduction

Communication between networked devices requires the following parameters to be configured:

- Network parameters

  Network parameters identify the network within the system configuration, for example, using a name.
- Interface parameters

  Interface parameters define specific properties of a component capable of communication. Addresses and transmission characteristics are set automatically and are consistent with the network parameters.

  Note

  Network parameters and interface parameters are usually set during networking such that communication can take place for numerous applications without the parameters having to be changed.

Procedure - checking or changing network parameters

Proceed as follows to check or change network parameters:

1. Enter the network view.
2. Select the subnet from the network view.

   You can see the network parameters in the "Properties" tab in the inspector window.
3. If necessary, check or modify the network parameters in the relevant group of parameters.

Procedure - checking or changing interface parameters

You can check and modify interface parameters in the network and device view.

Proceed as follows to check or change interface parameters:

1. Enter the network view or device view.
2. Select the interface.

   You can see the interface parameters in the "Properties" tab in the inspector window.
3. If necessary, check or modify the interface parameters in the relevant group of parameters.

Changing networkings

Introduction

You can cancel an interface's network connection or assign it to another subnet of the same subnet type.

Consequences

Depending on the version, a distinction should be made between:

- Canceling a network connection for an interface

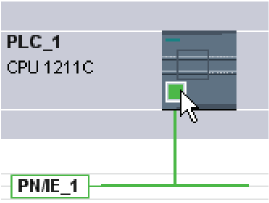
  The configured parameters for the interface remain unchanged.
- Assigning a network connection to another subnet

  If the addresses in the assigned subnet are not unique, in other words, they already exist, they will be changed automatically to make them unique.

Procedure - disconnecting from a network

Proceed as follows to cancel the network connection for an interface:

1. Select the networked interface.



2. Select the "Disconnect from subnet" command in the shortcut menu of the interface.

The network connection is deleted, the interface addresses are, however, not changed.

Configured connections are retained; however these connections are marked red in the connection table because they are not networked. Specified connections remain specified.

See also

Networking devices in the network view (Page 349)

Copying, cutting or deleting subnets

Introduction

You can copy subnets as individual objects or copy them along with networked devices or other networks.

For example, you can create complex configurations to be used more than once in different variants within the project with no additional effort.

Effects on the copied subnet

Properties that have to be assigned explicitly within a project are re-assigned appropriately when the copied objects are copied.

For subnets, this means: The subnet ID and name are re-assigned to the copied subnet.

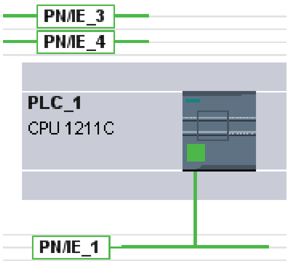
The configured properties are adopted in the copied subnet.

Procedure - copying a subnet

Proceed as follows to copy one or more subnets:

1. Select one or more subnets.
2. In the shortcut menu, select the "Copy" command.
3. Select the "Paste" command in the shortcut menu.

The copied subnets are shown as "orphaned" subnets in the top part of the network view.



Procedure - copying a subnet with connected devices

To copy one or more subnets with networked devices, proceed as follows:

1. Select one or more subnets with the connected devices, for example by drawing a lasso around them.
2. In the shortcut menu, select the "Copy" command.
3. Select the "Paste" command in the shortcut menu.

Complete copies of the subnets and connected devices are created.

Configured connections are adopted and remain within the copied devices. Connections to devices that have not been copied are interrupted and become unspecified.

MPI network configuration

Allocating MPI addresses

Note the following for devices with an MPI interface: All devices of a subnet must have different device addresses.

CPUs with MPI address ship with the default MPI address 2. Since you can only use this address once in the MPI subnet, you will have to change the default address in all other CPUs.

The following applies to devices with the order no. 6ES7 3xx-xxxxx-0AB0:

When planning the MPI addresses for several CPUs, you have to fill "MPI address gaps" for FMs and CPs with separate MPI addresses to prevent addresses being assigned twice.

Only when all the modules in a subnet have different addresses and your actual structure matches that of the network configuration produced, should you load the settings across the network.

Rules for assigning the MPI address

- Allocate the MPI addresses in ascending order.
- Reserve MPI address 0 for a programming device.
- You can link up to 126 (addressable) devices with one another in an MPI subnet; up to 8 devices at a transfer speed of 19.2 KB/s.
- All MPI addresses in an MPI subnet must be different.

You will find other rules relating to the structure of a network in the manuals for setting up automation systems.

PROFIBUS network configuration

PROFIBUS addresses

Rules for the network configuration

All the devices in a subnet must have a different PROFIBUS address.

Only when all the modules in a subnet have different addresses and your actual structure matches that of the network configuration produced, should you load the settings across the network.

You can connect devices to the PROFIBUS subnet that communicate via configured connections or that belong to a PROFIBUS DP master system.

You can find more information on configuring a DP master system in the following sections.

Requirements

The 121xC CPU is PROFIBUS compatible as of firmware version 2.0.

Rules for assigning PROFIBUS addresses

- Allocate the PROFIBUS addresses in ascending order.
- Reserve the PROFIBUS address "0" for a programming device.
- Allocate a unique PROFIBUS address between 0 and 126 for each device on the PROFIBUS network and/or for each DP master and each DP slave in the PROFIBUS network.
- There are modules with which the smallest address that can be set has to be greater than 1.
- All PROFIBUS addresses of a PROFIBUS subnet must be different.

You will find additional rules relating to the structure of a network in the manuals for setting up automation systems, for example SIMATIC S7-1200.

Note

PROFIBUS address "0"

Reserve PROFIBUS address "0" for a programming device that you will briefly connect up to the PROFIBUS network at a later date for servicing.

See also

What you need to know about PROFIBUS bus parameters (Page 359)

What you need to know about PROFIBUS bus parameters

Matching parameters to one another

The PROFIBUS subnet will only function without problem if the parameters for the bus profile are matched to one another. You should therefore only change the default values if you are familiar with how to configure the bus profile for PROFIBUS.

Note

It may be possible for the bus parameters to be adjusted depending on the bus profile. If the bus parameters cannot be adjusted, they are grayed out. The offline values of the bus parameters are always shown even if you are online and linked to the target system.

The parameters shown apply to the entire PROFIBUS subnet and are briefly explained below.

Activating cyclic distribution of the bus parameters

If the "Enable cyclic distribution of the bus parameters" check box is selected under "Bus parameters" while PROFIBUS subnet is selected in the Inspector window, the bus parameters are transferred cyclically during operation by the modules that support this function. This allows a programming device, for example, to be easily connected to the PROFIBUS in runtime.

You should deactivate this function:

- For a heterogeneous PROFIBUS subnet (or more accurately: when external devices are connected whose protocol uses the DSAP 63 for Multicast)
- When in constant bus cycle time mode (minimize bus cycle)

Bus parameters for the bus profile of PROFIBUS subnets

Bus parameter	Adjustable?	Limit values
Tslot_Init	Yes	Max. Tsdr + 15 <= Tslot_init <= 16.383 t_bit
Max. Tsdr	Yes	35 + 2*Tset + Tqui <= Max. Tsdr <= 1.023 t_bit
Min. Tsdr	Yes	11 t_bit <= Min. Tsdr <= MIN(255 t_bit, ... Max. Tsdr - 1, 34 + 2*Tset + Tqui)
Tset	Yes	1 t_bit <= Tset <= 494 t_bit
Tqui	Yes	0 t_bit <= Tqui <= MIN(31 t_bit, Min. Tsdr - 1)
GAP factor	Yes	1 <= GAP factor <= 100
Retry limit	Yes	1 <= Retry limit <= 15
Tslot	No	---
Tid2	No	Tid2 = Max. Tsdr
Trdy	No	Trdy = Min. Tsdr
Tid1	No	Tid1 = 35 + 2*Tset + Tqui
Ttr	Yes	256 t_bit <= Ttr <= 16.777.960 t_bit
Ttr typical	No	This time is provided for information only and is not transferred to the nodes.
Response monitoring		10 ms <= response monitoring (watchdog) <= 650 s

If you want to create a customized bus profile, we recommend the following settings:

- Minimum target rotation time (Ttr) = 5000 x HSA (highest PROFIBUS address)
- Minimum response monitoring (watchdog) = 6250 x HSA

Recalculating

You can use the "Recalculate" button to recalculate the parameters.

See also

PROFIBUS addresses (Page 358)

Description of the bus parameters (Page 361)

Description of the bus parameters

Detailed description of PROFIBUS bus parameters

Bus parameter	Meaning
Tslot_Init	The wait-for-reception (slot time) defines the maximum time the sender will wait to receive a response from the addressed partner. If the influence of the line components on message frame run times is entered in the "Cable Configuration" parameter group, these components must also be taken into account. The component is added to the specified Tslot_Init and the total used as Tslot.
Max. Tsdr	The maximum protocol processing time defines the latest time by which the responding node should have replied.
Min. Tsdr	The minimum protocol processing time defines the earliest time by which the responding node may reply.
Tset	The trigger time is the time which may lapse between the reception of a data message frame and the response to it in the node.
Tqui	The modulator quiet time is the time which a sending node needs after the end of the message frame to switch from sending to receiving.
GAP factor	The GAP update factor defines the number of token rotations after which a newly added, active node can be added to the logical token ring.
Retry limit	This parameter defines the maximum number of attempts (message frame repeats) made to reach a node.
Tslot	The wait-for-reception time (slot time) defines the maximum time the sender will wait to receive a response from the addressed partner. If the influence of the bus design components on message frame run times is entered in the "Cable Configuration" parameter group, these components must also be taken into account. The component is added to the specified Tslot_Init and the total used as Tslot.
Tid2	Idle time 2 defines the earliest time by which a sending node may send the next message frame after sending a message frame that has not been acknowledged.
Trdy	The ready time specifies the earliest time by which a sending node may receive a response message frame.
Tid1	Idle time 1 defines the earliest time by which a sending node may send the next message frame after receiving a response.
Ttr	The target rotation time is the maximum time made available for a token rotation. During this time, all active nodes (DP masters etc.) have the right to send (token) once. The difference between the target rotation time and the actual token holding time of a node determines how much time is left over for the other active nodes (programming device, other DP masters etc.) to send message frames.
Ttr typical	The typical data cycle time is the average response time on the bus if all configured slaves are exchanging data with the DP master. None of the slaves report diagnostics and there is no extra message frame traffic with programming devices or other active nodes etc. on the bus.
Response monitoring	The response monitoring time is only needed for PROFIBUS-DP bus systems. It defines the latest time by which a DP slave has to be addressed by its DP master with a new data message frame. If this does not happen, the DP slave assumes that the DP master has failed and resets its outputs to a secure mode.

See also

What you need to know about PROFIBUS bus parameters (Page 359)

Bus profiles with PROFIBUS

Introduction

Depending on the device types connected and protocols used on the PROFIBUS, different profiles are available. The profiles differ in terms of the setting options and calculation of bus parameters. The profiles are explained below.

Devices with different profiles on the same PROFIBUS subnet

The PROFIBUS subnet only functions without problem if the bus parameters of all devices have the same values. For example, if both DP and FMS services are being used on a subnet, the "slower" sets of bus parameters must always be set for all devices, i.e. even the "Universal (DP/FMS)" profile for DP devices.

Profiles and transmission rates

Profiles	Supported transmission speeds in Kbits/s
DP	9.6 19.2 45.45 93.75 187.5 500 1500 3000 6000 12000
Standard	9.6 19.2 45.45 93.75 187.5 500 1500 3000 6000 12000
Universal (DP-FMS)	9.6 19.2 93.75 187.5 500 1500
Customized	9.6 19.2 45.45 93.75 187.5 500 1500 3000 6000 12000

Meaning of profiles

Profile	Meaning
DP	Select the "DP" bus profile when the only devices connected to the PROFIBUS subnet are those which satisfy the requirements of standard EN 50170 Volume 2/3, Part 8-2 PROFIBUS. The bus parameter setting is optimized on these devices. This includes devices with DP master and DP slave interfaces of the SIMATIC S7 and distributed I/Os of other manufacturers.
Standard	Compared to the "DP" profile, the "Standard" profile also offers scope for devices of another project or devices which have not been configured here to be taken into account when calculating the bus parameters. The bus parameters are then calculated following a simple, non-optimized algorithm.
Universal (DP/FMS)	Select the "Universal (DP/FMS)" bus profile when individual devices on the PROFIBUS subnet use the PROFIBUS-FMS service. This includes the following devices for example: • CP 343-5 (SIMATIC S7) • PROFIBUS-FMS devices of other manufacturers As with the "Standard" profile, this profile allows you to take other devices into account when calculating the bus parameters.
Customized	The PROFIBUS subnet will only function without problem if the parameters for the bus profile are matched to one another. Select the "Customized" profile when none of the usual profiles "match" a PROFIBUS device and you need to adapt the bus parameters to your special structure. Information on this can be found in the documentation for the PROFIBUS device. You should only change the default values if you are familiar with how to configure the bus profile for PROFIBUS. Not all combinations that can be theoretically set can be used even with this bus profile. The PROFIBUS standard specifies several parameter limits that depend on other parameters. For example, a responder must not respond (Min Tsdr) before the initiator can receive the message frame (Trdy). These standard specifications are also checked in the "Customized" profile. Tip: The bus parameters last valid on the PROFIBUS subnet are always automatically set as customized. For example, if the "DP" bus profile was valid for the subnet, then the bus parameters for "DP" are set in the "Customized" bus profile. The parameters can be modified on this basis. The monitoring times are not automatically recalculated so that you do not put at risk the consistency of set values, for example with configurations in other configuration tools without realizing that you have done so. You can also have the Ttr monitoring times and target rotation time calculated on the basis of parameters you have set: Click here on the "Recalculate" button.

Note

Both mono-master mode and multi-master mode are possible with all PROFIBUS profiles.

What you need to know about PROFIBUS line configuration

Cable configuration and bus parameters

Information regarding the cable configuration can be taken into consideration when calculating the bus parameters. For this purpose, you must select the "Consider cable configuration" check box in the properties for the PROFIBUS subnet.

The remaining information then depends on the type of cable used; the following settings are available:

- Copper cable
- Fiber-optic cable/optical ring

PROFIBUS line configuration, optical ring

The calculation depends on the OLM types used. The selection is made by activating the check box (multiple activation is possible, at least one OLM type must be selected):

- OLM/P12
- OLM/G12
- OLM/G12-EEC
- OLM/G12-1300

The following bus parameter adjustments are made:

- Configuration of a node not present

Note

The following restrictions apply to optical rings, even for passive nodes (DP slaves for example):

A maximum of HSA-1 nodes may be connected to the PROFIBUS network. With an HSA of 126, addresses 126 and 125 must not be used. A maximum of 125 nodes are possible on the bus (No. 0 to 124).

If an HSA is less than or equal to 125, the addresses HSA and greater may not be used. The address HSA-1 may not be used.

- Increase the retry value to 3
- Setting of minimum slot time needed for ring mode

Note

Short slot time values are needed for OLM/P12, average ones for OLM/G12 and OLM/G12-EEC and long ones for OLM/G12-1300. This results in high performance for small network lengths and average to low performance for average to large network lengths.

PROFIBUS communication load

Communication load - allowing for additional network stations

The bus parameters depend on the volume of communication between the active network nodes. There are differences between cyclic communication (DP) and connection-based, acyclic communication (S7 communication, Send/Receive (FDL), FMS). Unlike DP, the volume and size of communication tasks (communication load) depends on the user program. For this reason, the communication load cannot always be calculated automatically.

To calculate the bus times you can define a network configuration in the "Additional network stations" parameter group that differs from the network configuration.

Taking the profile into account

The network configuration can be defined for the "Standard", "Universal (DP/FMS)", and "User-defined" profiles. Parameters cannot be entered in the "Additional network stations" parameter group for the "DP" profile.

Quantification of the communication load

The following settings are available in order to make allowance for the communication load.

- Information regarding the number of unconfigured network stations;
- Information on the communication load resulting from the user programs for FDL or S7 communication. Here you can selected from the following settings:
  - Low

    Typically used for DP, no great data communication apart from DP.
  - Medium

    Typically used for mixed operations featuring DP and other communication services (such as for S7 communication), when DP has strict time requirements and for average acyclic volumes of communication.
  - High

    For mixed operations featuring DP and other communication services (such as for S7 communication), when DP has loose time requirements and for high acyclic volumes of communication.

Configuring Industrial Ethernet

Rules for the network configuration

The Ethernet interfaces of the devices have a default IP address that you can change.

IP address

The IP parameters are visible if the module capable of communication supports the TCP/IP protocol. This is usually the case for all Ethernet modules.

The IP address consists of 4 decimal figures in the range of 0 to 255. The decimal figures are separated from one another by a dot.

Example: 140.80.0.2

The IP address consists of:

- Address of the (sub) net
- Address of the node (generally also called host or network node)

Subnet mask

The subnet mask splits these two addresses. It determines which part of the IP address addresses the network and which part of the IP address addresses the node.

The set bits of the subnet mask determine the network part of the IP address.

Example:

Subnet mask: 255.255.0.0 = 11111111.11111111.00000000.00000000

In the example given for the above IP address, the subnet mask shown here has the following meaning:

The first 2 bytes of the IP address identify the subnet - i.e. 140.80. The last two bytes address the node, thus 0.2.

It is generally true that:

- The network address results from AND linking the IP address and subnet mask.
- The device address results from AND NOT linking the IP address and subnet mask.

Relation between IP address and default subnet mask

An agreement exists relating to the assignment of IP address ranges and so-called "Default subnet masks". The first decimal number (from the left) in the IP address determines the structure of the default subnet mask. It determines the number of "1" values (binary) as follows:

IP address (decimal)	IP address (binary)	Address class	Default subnet mask
0 to 126	0xxxxxxx.xxxxxxxx....	A	255.0.0.0
128 to 191	10xxxxxx.xxxxxxxx...	B	255.255.0.0
192 to 223	110xxxxx.xxxxxxxx...	C	255.255.255.0

Note

Range of values for the first decimal point

A value of between 224 and 255 is also possible for the first decimal number of the IP address (address class D etc). This is, however, not recommended because there is no address check for these values.

Masking other subnets

You can use the subnet mask to add further structures and form "private" subnets for a subnet that is assigned one of the address classes A, B or C. This is done by setting other lower points of the subnet mask to "1". For each bit set to "1", the number of "private" networks doubles and the number of nodes they contain is halved. Externally, the network functions like an individual network as it did previously.

Example:

You have a subnet of address class B (for example IP address 129.80.xxx.xxx) and change the default subnet mask as follows:

Masks	Decimal	Binary
Default subnet mask	255.255.0.0	11111111.11111111.00000000.00000000
Subnet mask	255.255.128.0	11111111.11111111.10000000.00000000

Result:

All nodes with addresses between 129.80.001.xxx and 129.80.127.xxx are on one subnet, all nodes with addresses between 129.80.128.xxx and 129.80.255.xxx are on another subnet.

Router

The task of the routers is to connect the subnets. If an IP datagram is to be sent to another network, it first has to be conveyed to a router. To make this possible, you have to enter the address of the router for each node in the subnet.

The IP address of a node in the subnet and the address of the router may only differ at the points at which there is a "0" in the subnet mask.

Network configuration of AS interface

AS-Interface consists of an AS-i master and AS-i slaves connected to each other over an AS-i subnet.

Rules for AS interface network configuration

All the nodes in an AS-i subnet must have a different AS-i node address.

You should only load the settings over the network when all the modules in a subnet have different addresses and when the actual structure matches that of the network configuration you have created.

One AS-i master and up to 31 AS-i slaves can be operated on one AS-i subnet.

For more information on configuring an AS-Interface with an AS-i master and AS-i slaves, refer to the section on AS-Interface and the documentation of the AS-i master modules.

8.1.3.2 Communication via connections

Working with connections

S7 connection

Introduction to configuring connections

Definition

A connection defines a logical assignment of two communication partners in order to undertake communication services. A connection defines the following:

- Communication partner involved
- Type of connection (for example, S7 connection)
- Special properties (e.g., whether a connection is established permanently or whether it is established and terminated dynamically in the user program, and whether status messages are to be sent)
- Connection path

What you need to know about connection configuration

During connection configuration, a local connection name is assigned for an S7 connection as a unique local identification.

In the network view, a "Connections" tab is displayed in addition to the "Network overview" tab. This tab contains the connection table. A row in this connection table represents a configured connection from the viewpoint of the local communication partner with its properties, for example, between two S7-1200 CPUs.

What you need to know about using connection resources

Introduction

Each connection requires connection resources for the end point and/or transition point on the devices involved. The number of connection resources is device-specific.

If all the connection resources of a communication partner are assigned, no new connections can be established. This situation is apparent when a newly created connection in the connection table has a red background. The configuration is then inconsistent and cannot be compiled.

S7 connections

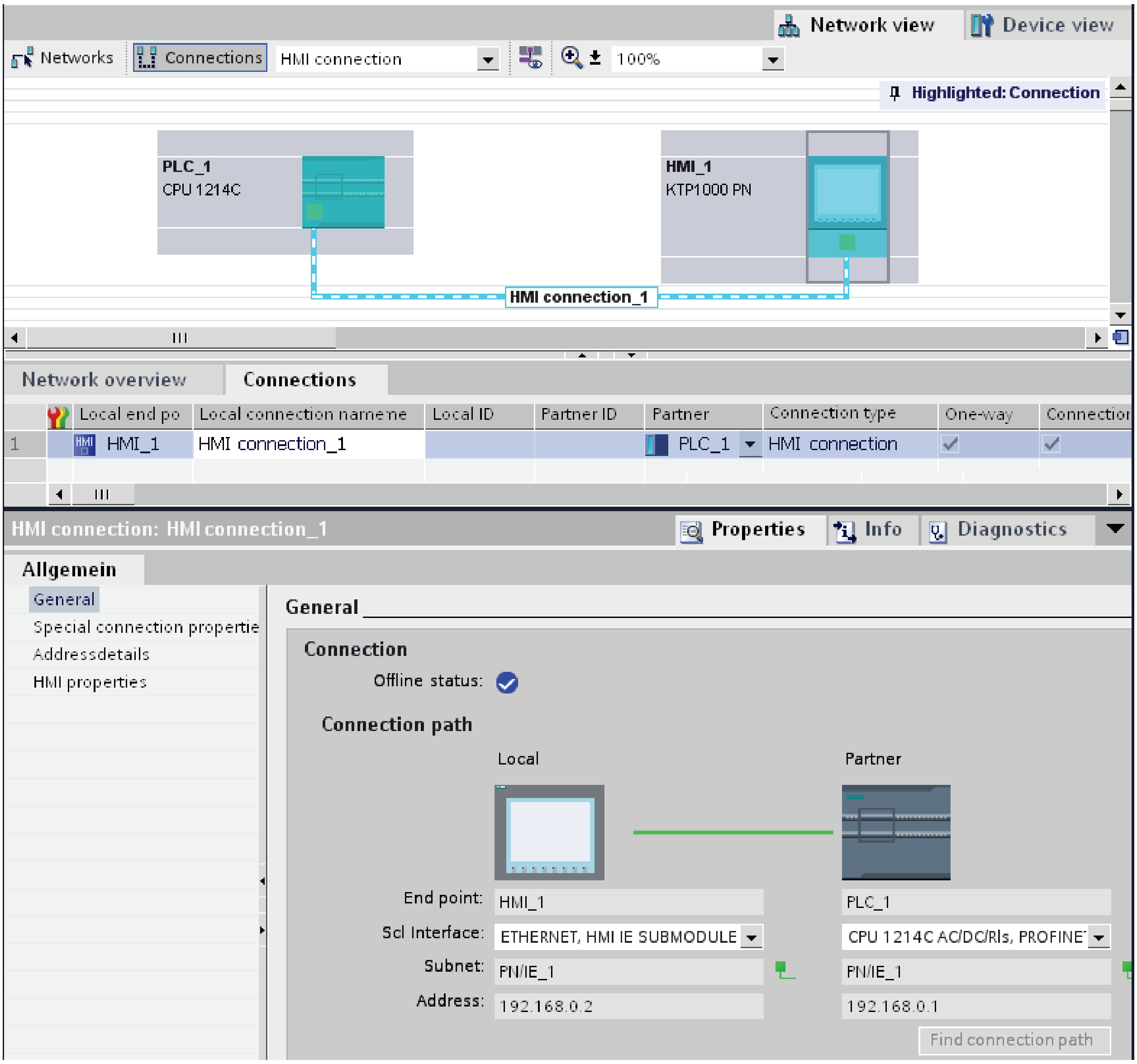
In the case of S7 connections via the PN interface, one connection resource per S7 connection is assigned for the endpoint for the S7-1200 CPU. One connection resource is also required for the connection partner.

You can find an overview of available and assigned connection resources for selected S7-1200 CPU in the Inspector window at "Properties > Connection Resources"

Views containing information about the configured connections

The views described below will provide you with comprehensive access to all the information and functions regarding configuring and checking communication connections.

- Connection display in the network view
- Connection table
- "Properties" tab for a connection in the inspector window



	Local end po	Local connection name ne	Local ID	Partner ID	Partner	Connection type	One-way	Connectior
1	HMI_1	HMI connection_1			PLC_1	HMI connection	✓	✓

HMI connection: HMI connection_1

General

Connection

Offline status: ✓

Connection path

	Local	Partner
End point:	HMI_1	PLC_1
Scl Interface:	ETHERNET, HMI IE SUBMODULE	CPU 1214C AC/DC/Rls, PROFINE
Subnet:	PN/IE_1	PN/IE_1
Address:	192.168.0.2	192.168.0.1

Find connection path

Benefits

The information shown in these views are always up-to-date in terms of the current user actions. This means:

- The connection table displays all connections created.
- If you have selected a connection in the connection table:
  - When connection mode is enabled, the connection path is highlighted in the network view.
  - The "Properties" tab in the Inspector window displays the parameters of this connection.

The connection table

The connection table offers the following functions:

- List of all connections in the project
- Selection of a connection and display of connections associated with it in the network view (when connection mode is enabled)
- Changing of connection partners
- Display showing status information

"Properties" tab for a connection in the inspector window

The properties dialog has the following meaning:

- Display for connection parameters
- Display of connection path
- Subsequent specification of connections using the "Find connection path" button

Creating a new connection

Creating a connection - alternatives

You have the following options for creating a connection in the network view:

- Graphic connection configuration
- Interactive connection configuration

You'll find the individual steps for this in the following chapters.

Requirement and result

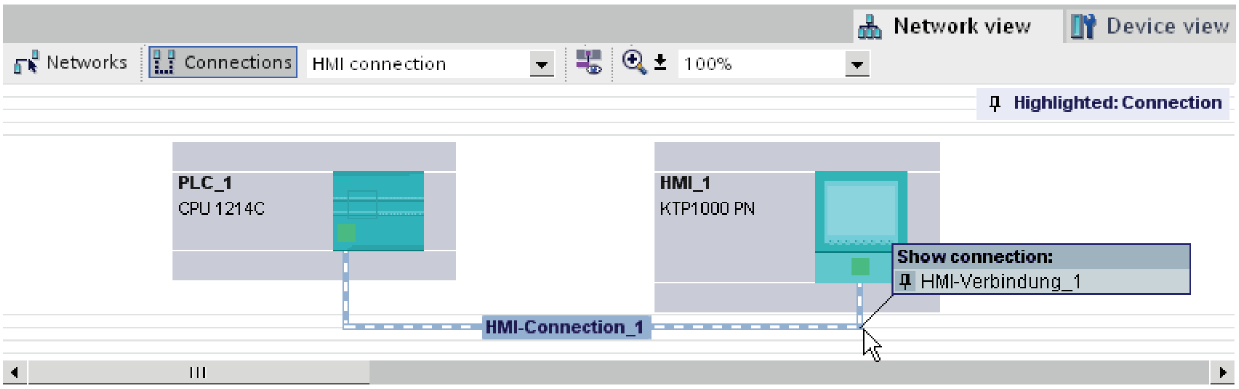
In the network view, you have add devices between which the connections should be configured.

Specifying a connection

If both partners for the connection type selected are networked on the same network, use the graphic or interactive selection of both communication partners to create a fully specified connection.

This connection is entered automatically in the connection table of the S7-1200 CPU. A local connection name is assigned for this connection.

The following schematic shows a configured connection with a networked device:



Creating a new connection graphically

Graphically configuring connections

In the case of graphic connection configuration, the connection path is automatically specified provided interfaces and resources are available. Select the devices to be connected in the current configuration.

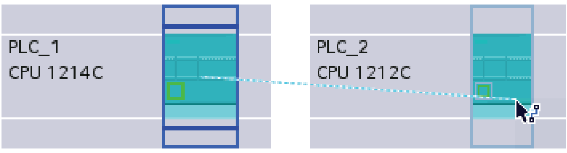
Automatically determining connection path

To create a connection graphically, follow these steps:

1. Click the "Connections" button.

   This step activates the connection mode: You can now select the connection type you want. You will see this from the following:
   The devices that can be used for the connection type selected in your project are color-highlighted in the network view.

2. Hold down the mouse button and drag the mouse pointer from the device from which the connection will originate to the device at which the connection ends.

3. Release the mouse button over the destination device to create the connection between the two devices.

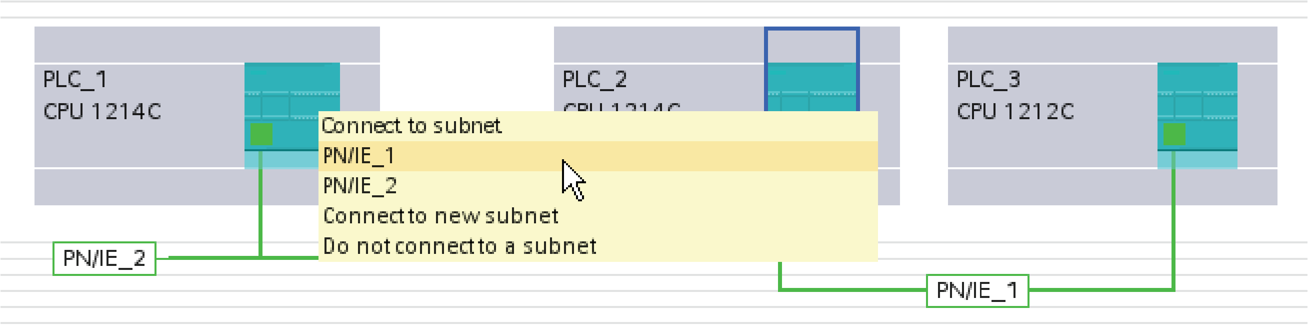
Result

- A specified connection is created.
- The connection path is highlighted.
- The connection is entered in the connection table.

Configuring a connection when there is no or no clear network assignment

Any networking not in place will if possible be created automatically when a connection is created. A query will be made upon completion of connection configuration if unique network assignment is not possible. You will be able to choose from the existing subnets.

Example below: A query is made upon creation of a connection between stations PLC_1 and PLC_2, which are not yet networked.



Interactively creating a new connection

Interactively configuring connections

Define the local device and its connection partners.

Procedure

Proceed as follows to interactively create a connection:

1. Select the "Add new connection" command in the shortcut menu of a connection partner for which you want to create a connection.
   The "Create new connection" dialog appears.
2. Select the partner endpoint.
   In the right pane of the dialog, a possible connection path fitting the selected endpoint is displayed, if available. Incomplete paths, for example, for a non-specified CPU, are marked by an exclamation mark on a red background.
3. To accept the configured connection and to configure additional connections to other endpoints, click "Add".
   To close the dialog, click "OK".

Working in the network view

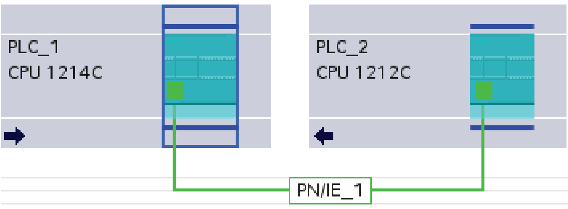
Highlighting connection path and partner in the network view

To display the connection partners for all or certain connection types in the network view, proceed as follows:

1. Click the "Connections" button.

2. Select the S7-CPU for which you want to display the connection partners in the network view and then select the "Highlight connection partners" command in the shortcut menu.
3. Select "All connection partners" in the following menu.

   The local device and the CPUs of the target devices are selected. The local connection partner shows an arrow pointing right and the remote connection partners show an arrow pointing left.
4. To open a list with information on the target devices, click the arrow of the local device. This additional function is useful in complex network configurations in which some devices are not visible.

Note

You can display one of the connection partners which cannot be seen in the current display range of the network view. Click on the communication partner in the list that appears. Result: The display is moved such that the connection partner becomes visible.

Working with the connection table

Basic functions for tables

The connection table supports the following basic functions for editing a table:

- Changing column width
- Displaying the meaning of a column, a row or cell using tooltips.

Changing column width

To adjust the width of a column to the content so that all texts in the lines are legible, follow these steps:

1. Position the cursor in the header of the connection table to the right of the column that you want to optimize until the cursor changes its shape to two parallel lines (as if you wanted to change the width of the column by dragging it with the cursor).
2. Double click on this point.

or

1. Open the shortcut menu on the header of the table.
2. Click on
   – "Optimize column width" or
   – "Optimize width of all columns".

For columns that are too narrow, the complete content of specific fields is shown when you pause with the cursor on the respective field.

Show / hide columns

You can use the shortcut menu of the header of the connection table to control the display of the various table columns. The shortcut menu entry "Show/hide columns" provides you with an overview of the available columns. Use the check box to control whether columns are shown or hidden.

If you want to store the layout, width and visibility of the table columns, click on the "Remember layout" function in the top right-hand of the network view.

Symbol	Meaning
▤	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Using cursor keys to move within the connection table

You can use the UP and DOWN cursor keys to select a connection from the connection table; the selected connection is marked and is shown highlighted in the network view.

Changing properties of connection

You can directly edit some of the parameters displayed in the connection table. The name of the connection can, for example, only be changed in the connection table.

Changing connection partners

You can change the connection partner of a connection as follows:

1. Select the connection.
2. Select the new connection partner from the open drop-down list in the "Partner" column.

Deleting connections

You can delete configured connections via the network view or the connection table.

In the network view you can delete a highlighted connection. In the connection table you can delete one or more connections.

Procedure

To delete a connection, proceed as follows:

1. Select the connection to be deleted:
   - In the network view: Select the connection to be deleted.
   - In the connection table: Select the rows of the connections to be deleted (multiple selection possible).
2. Open the shortcut menu with a right mouse click.
3. Select the "Delete" command.

The selected connection will be completely deleted.

Copying connections

Introduction

Connections are not copied singly but always in context along with the project or the device.

You can copy:

- Entire projects
- One or more devices within a project or from one project to another

Copying a project

When you copy a project all configured connections will also be copied. No settings whatsoever are required for the copied connections because the connections remain consistent.

Copying devices

If you copy devices for which connections have been configured, the connections are copied as well. To complete the connection path, you must still finalize the networking.

An S7-1200 CPU with a V.10 firmware is merely a server for connections and has no connection configuration itself. Consequently, no connections are copied along with it when an S7-1200 CPU with a V1.0 firmware is copied.

Inconsistent connections - connections without assignment

With an inconsistent connection the structure of the connection data is destroyed or the connection is not functional in the project context.

Inconsistent connections cannot be compiled and loaded - operation is not possible with such a connection.

In the connection table inconsistent connections are marked in red.

Possible causes for inconsistent connections

- Deletion or change of the hardware configuration.
- Missing interface network links in the project, which are necessary for a connection.
- Connection resources are exceeded
- Connections to an unspecified connection partner without partner address information.

Detailed information regarding the reasons for the inconsistency can be found in the "Compile" tab following compilation (Edit > Compile).

Remedies

To assign a closed connection path to an existing open connection path, expand the device configuration in such a way that the interfaces required for the connection type are available for both partners. At "Properties > General > Interface" in the Inspector window, you can use the "Find connection path" button to create a connection to an existing partner.

S7 connection - general settings

General connection parameters

General connection parameters are displayed in the "General" parameter group under the properties of the connection; these connection parameters identify the local connection end point.

Here, you can assign the connection path and specify all aspects of the connection partner.

Local ID

The local ID of the module from which the connection is viewed is displayed here (local partner). You can change the local ID. You may need to do this if you have already programmed communication function blocks, and you want to use the local ID specified in those function blocks for the connection.

Special connection properties

Display of connection properties (can be modified depending on the components used):

- One-way

  One-way means that the connection partner functions as a server on this connection and cannot send or receive actively.

- Active connection establishment

  In the case of one-way connection, for example with a S7-1200 CPU (firmware version V1.0), a connection partner can only be provided for the active connection establishment. In the case of a two-way connection you can set which connection partner will assume the active role.

- Sending operating mode messages

  Indicates whether or not the local partner sends operating mode messages to the connection partner.

Address details

Displaying address details of the S7 connection. With an unspecified partner, the values for the rack and slot can be changed. All other values are obtained from the current configuration and cannot be changed.

S7 connection - address details

Meaning

The address details show the end points of the connection and can localize these via the specification of rack and slot.

When a connection is established, the connection-specific resources of a module are assigned specifically to this connection. This assignment requires that the connection resource can be addressed. The TSAP (Transport Service Access Point) is, as it were, the address of the resource that is formed with the help of the connection resource or, in the case of S7-1200 CPUs (firmware V2.0 or higher) with the SIMATIC-ACC (SIMATIC Application Controlled Communication).

Configuration of TSAP for S7-1200

- For S7-1200 CPU (firmare V2.0 or higher):

  "SIMATIC-ACC"<nnn><mm>

  nnn = Local ID

  mm = any value

- For S7-1200 CPU (firmare V1.0):

  <xx>.<yz>

  xx = Number of the connection resource

  y = Rack number

  z = Slot number

TSAP structure, dependent on partner

The configuration of the TSAP for S7-1200 CPUs is dependent on the respective firmware and on the remote connection partner. When a S7-1200 CPU is connected with a S7-300/400 CPU, a S7-1200 CPU also uses a TSAP configuration that includes the connection resource.

See the following examples for TSAPs of various connection configurations

- Connection between two S7-1200 CPUs (both with firmware V2.0):
  - S7-1200 CPU "A" with firmware V2.0 and local ID 100:

    TSAP: SIMATIC-ACC10001
  - S7-1200 CPU "B" with firmware V2.0 and local ID 5AE:

    TSAP: SIMATIC-ACC5AE01
- Connection between two S7-1200 CPUs (firmware V2.0 and V1.0):
  - S7-1200 CPU with firmware V2.0 and local ID 1FF:

    TSAP: SIMATIC-ACC1FF01
  - S7-1200 CPU with firmware V1.0 (rack 0, slot 1, connection resource 03):

    TSAP: 03.01
- Connection between S7-1200 CPUs (firmware V2.0) and S7-300/400 CPU:
  - S7-1200 CPU with firmware V2.0 (rack 0, slot 1, connection resource 12):

    TSAP: 12.01
  - S7-300/400 CPU (rack 0, slot 2, connection resource 11):

    TSAP: 11.02

HMI connection

Introduction to configuring connections

Definition

A connection defines a logical assignment of two communication partners in order to undertake communication services. A connection defines the following:

- Communication partner involved
- Type of connection (e.g., HMI connection)
- Special properties (e.g., whether a connection is established permanently or whether it is established and terminated dynamically in the user program, and whether status messages are to be sent)
- Connection path

What you need to know about connection configuration

During connection configuration, a local connection name is assigned for an HMI connection as a unique local identification.

In the network view, a "Connections" tab is displayed in addition to the "Network overview" tab. This tab contains the connection table. A line in this connection table represents a configured connection, e.g., between an HMI device and PLC, along with its properties.

What you need to know about using connection resources

Introduction

Each connection requires connection resources for the end point and/or transition point on the devices involved. The number of connection resources is device-specific.

If all the connection resources of a communication partner are assigned, no new connections can be established. This situation is apparent when a newly created connection in the connection table has a red background. The configuration is then inconsistent and cannot be compiled.

HMI connections

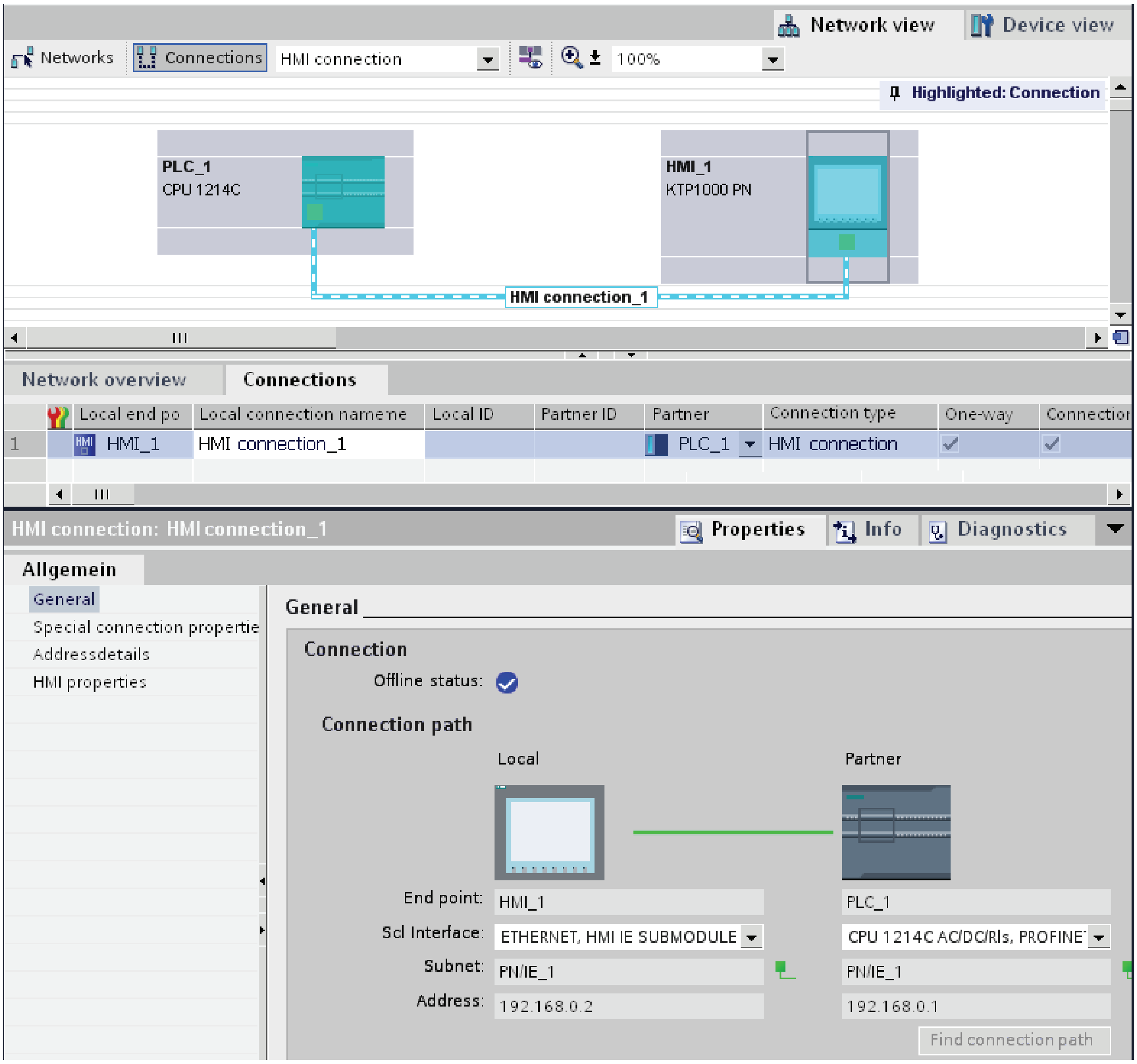
For HMI connections via the **integrated** PN interface, one connection resource for the endpoint per HMI connection is occupied for the HMI device.

One connection resource is also required for the connection partner (PLC).

Views containing information about the configured connections

The views described below will provide you with comprehensive access to all the information and functions regarding configuring and checking communication connections.

- Connection display in the network view
- Connection table
- "Properties" tab for a connection in the inspector window



	Local end po	Local connection name ne	Local ID	Partner ID	Partner	Connection type	One-way	Connectio
1	HMI_1	HMI connection_1			PLC_1	HMI connection	✓	✓

HMI connection: HMI connection_1

General

Connection

Offline status: ✓

Connection path

	Local	Partner
End point:	HMI_1	PLC_1
Scl Interface:	ETHERNET, HMI IE SUBMODULE	CPU 1214C AC/DC/Rls, PROFINE
Subnet:	PN/IE_1	PN/IE_1
Address:	192.168.0.2	192.168.0.1

Benefits

The information shown in these views are always up-to-date in terms of the current user actions. This means:

- The connection table displays all connections created.
- If you have selected a connection in the connection table:
  - You will graphically see the connection path in the network view.
  - The "Properties" tab in the Inspector window displays the parameters of this connection.

The connection table

The connection table offers the following functions:

- List of all connections in the project
- Selection of a connection and display of connections associated with it in the network view
- Changing of connection partners
- Display showing status information

"Properties" tab for a connection in the inspector window

The properties dialog has the following meaning:

- Display for connection parameters
- Display of connection path
- Subsequent specification of connections using the "Find connection path" button

Creating a new connection

Creating a connection - alternatives

You have the following options for creating a connection in the network view:

- Graphic connection configuration
- Interactive connection configuration

You'll find the individual steps for this in the following chapters.

Requirement and result

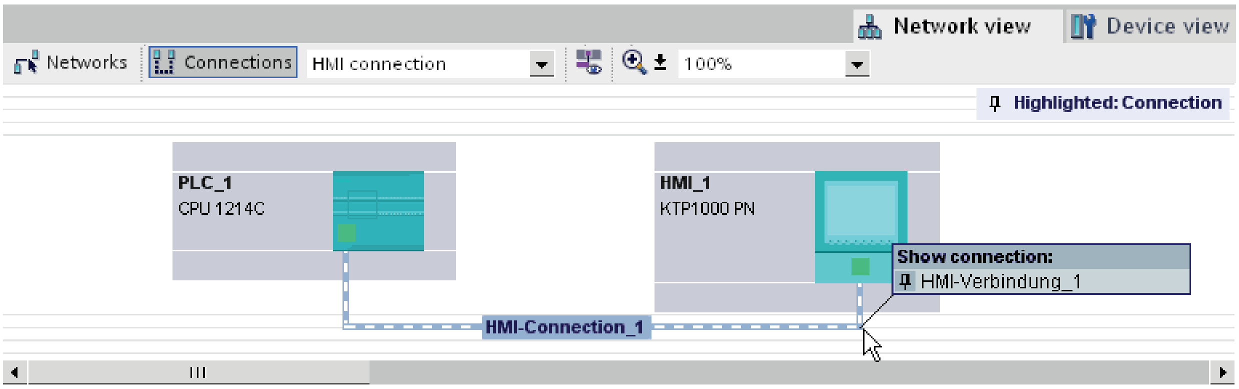
You have created the devices with CPUs and HMI devices between which you want to configure connections in the network view.

Specifying a connection

If both partners for the connection type selected are networked on the same network, use the graphic or interactive selection of both communication partners to create a fully specified connection.

This connection is entered automatically into the connection table of the HMI device. A local connection name is assigned for this connection.

The following schematic shows a configured connection with a networked device:



Creating a new connection graphically

Graphically configuring connections

When using the graphic connection configuration, if necessary the system asks you to define the connection path. Select the devices to be connected in the current configuration.

Automatically determining connection path

To create a connection graphically, follow these steps:

1. Click the "Connections" button.

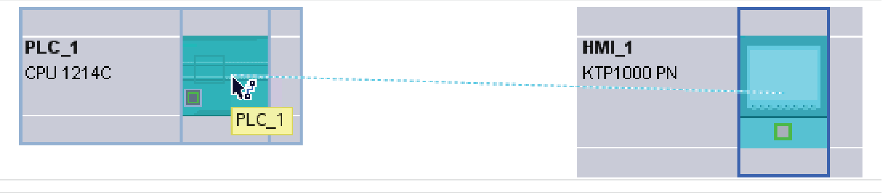
Connections

The connection mode for the connection type you have selected is then activated.

You will see this from the following:

The devices that can be used for the connection type selected in your project are color-highlighted in the network view.

2. Hold down the mouse button and drag the mouse pointer from the device from which the connection will originate to the device at which the connection ends.



3. Release the mouse button over the destination device to create the connection between the two devices.

Result

- A specified connection is created.
- The connection path is highlighted.
- The connection is entered in the connection table.

Interactively creating a new connection

Interactively configuring connections

Define the local device and its connection partners.

Procedure

Proceed as follows to interactively create a connection:

1. Select the "Create new connection" command in the shortcut menu of a connection partner for which you want to create a connection.

   The "Create new connection" dialog is opened.

2. Select the partner endpoint.

   In the right pane of the dialog, a possible connection path fitting the selected endpoint is displayed, if available. Incomplete paths, for example, for a non-specified CPU, are marked by an exclamation mark on a red background.

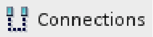
3. To close the dialog, click "OK".
   To accept the configured connection and to configure additional connections to other endpoints, click "Apply".

Working in the network view

Highlighting connection path and partner in the network view

To display the connection partners for all or certain connection types in the network view, proceed as follows:

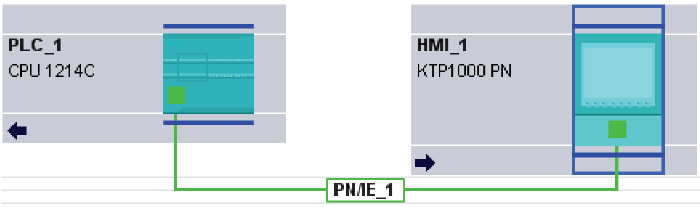
1. Click the "Connections" button.

2. Select the "Highlight connection partners" command in the shortcut menu for the HMI device whose connection partners you want to display in the network view.
3. Select "All connection partners" in the following menu.

   The local device and the CPUs of the target devices are selected. The local connection partner shows an arrow pointing right and the remote connection partners show an arrow pointing left.

4. To open a list with information on the target devices, click the arrow of the local device. This additional function is useful in complex network configurations in which some devices are not visible.



Note

You can display one of the connection partners which cannot be seen in the current display range of the network view. Click on the communication partner in the list that appears. Result: The display is moved such that the connection partner becomes visible.

See also

Creating a new connection graphically (Page 385)

Working with the connection table

Basic functions for tables

The connection table supports the following basic functions for editing a table:

- Changing column width
- Displaying the meaning of a column, a row or cell using tooltips.

Changing column width

To adjust the width of a column to the content so that all texts in the lines are legible, follow these steps:

1. Position the cursor in the header of the connection table to the right of the column that you want to optimize until the cursor changes its shape to two parallel lines (as if you wanted to change the width of the column by dragging it with the cursor).
2. Double click on this point.

or

1. Open the shortcut menu on the header of the table.
2. Click on
   - "Optimize column width" or
   - "Optimize width of all columns".

For columns that are too narrow, the complete content of specific fields is shown when you pause with the cursor on the respective field.

Show / hide columns

You can use the shortcut menu of the header of the connection table to control the display of the various table columns. The shortcut menu entry "Show/hide columns" provides you with an overview of the available columns. Use the check box to control whether columns are shown or hidden.

Using cursor keys to move within the connection table

You can use the UP and DOWN cursor keys to select a connection from the connection table; the selected connection is marked and is shown highlighted in the network view.

Changing properties of connection

You can directly edit the parameters displayed in the connection table in some cases. To change the name of a connection, you do not have to navigate to the Inspector window.

Changing connection partners

You can change the connection partner of a connection as follows:

1. Select the connection.
2. Select the new connection partner from the open drop-down list in the "Partner" column.

Deleting connections

You can delete configured connections via the network view or the connection table.

In the network view you can delete a highlighted connection. In the connection table you can delete one or more connections.

Procedure

To delete a connection, proceed as follows:

1. Select the connection to be deleted:
   - In the network view: Select the connection to be deleted.
   - In the connection table: Select the rows of the connections to be deleted (multiple selection possible).
2. Open the shortcut menu with a right mouse click.
3. Select the "Delete" command.

The selected connection will be completely deleted.

Copying connections

Introduction

Connections are not copied singly but always in context along with the project or the device.

You can copy:

- Entire projects
- One or more devices within a project or from one project to another

Copying a project

When you copy a project all configured connections will also be copied. No settings whatsoever are required for the copied connections because the connections remain consistent.

Copying devices

If you copy devices for which connections have been configured (HMI devices), the connections are copied as well. To complete the connection path, you must still finalize the networking.

An S7-1200 CPU with a V.10 firmware is only a server for connections and has no connection configuration itself. Consequently, no connections are copied along with it when an S7-1200 CPU with a V1.0 firmware is copied.

Inconsistent connections - connections without assignment

With an inconsistent connection the structure of the connection data is destroyed or the connection is not functional in the project context.

Inconsistent connections cannot be compiled and loaded - operation is not possible with such a connection.

In the connection table inconsistent connections are marked in red.

Possible causes for inconsistent connections

- Deletion or change of the hardware configuration.
- Missing interface network links in the project, which are necessary for a connection.
- Connection resources are exceeded
- Errors when backing up data due to insufficient memory
- Connections to an unspecified connection partner without partner address information.

Detailed information regarding the reasons for the inconsistency can be found in the "Compile" tab following compilation (Edit > Compile).

Remedies

If the connection cannot be repaired by opening the connection properties, changing them or undoing them in the configuration, then it may be necessary to delete the connection and re-create it.

HMI connection general settings

General connection parameters

General connection parameters are displayed in the "General" parameter group under the properties of the connection; these connection parameters identify the local connection end point.

Here, you can also assign the connection path and specify all aspects of the connection partner.

Special connection properties

Display of the connection properties (cannot be changed):

- Active connection establishment

  The connection establishment always starts from the HMI device. This option is selected by default if the address of the partner is specified.
- One-way

  One-way means that the connection partner functions as a server on this connection and cannot send or receive actively.
- Sending operating mode messages

  Not relevant for HMI devices.

Address details

Displaying address details of the HMI connection. With an unspecified partner, the values for the rack and slot can be changed. All other values are obtained from the current configuration and cannot be changed.

Miscellaneous

Display of the access points for the online connection between HMI device and connection partner.

Using Open User Communication

Basics of Open User Communication

Introduction

Open User Communication is the name of a program-controlled communications technique for communication via the integrated PN/IE interface of the CPU. Various connection types are available for this communications technique.

The main feature of Open User Communication is its high degree of flexibility in terms of the data structures transferred. This allows open data exchange with any communicating devices providing they support the connection types available here. Since this communication is controlled solely by instructions in the user program, event-driven connection establishment and termination is possible. Connections can also be modified by the user program during runtime.

For S7-1200 CPUs with an integrated PN/IE interface, the TCP, UDP and ISO-on-TCP connection types are available for Open User Communication. The communications partners can be two SIMATIC PLCs or a SIMATIC PLC and a suitable third-party device.

Instructions for Open User Communication

To create the connections, you have various instructions available after opening the program editor in the "Instructions > Extended instructions > Communication" task card:

- Compact instructions for sending or receiving data via integrated functions for establishing and terminating the connection:
  - TSEND_C (connection establishment/termination, sending)
  - TRCV_C (connection establishment/termination, receiving)
- Individual instructions for sending and receiving data or for establishing or terminating connections:
  - TCON (connection establishment)
  - TDISCON (connection termination)
  - TSEND (TCP or ISO-on-TCP: Sending)
  - TRCV (TCP or ISO-on-TCP: Receiving)
  - TUSEND (UDP: Sending)
  - TURCV (UDP: Receiving)

Connection establishment

For Open User Communication, instructions for establishing and terminating the connection must exist for both communications partners. One communications partner sends its data using TSEND, TUSEND or TSEND_C while the other communications partner receives the data using TRCV, TURCV or TRCV_C.

One of the communications partners starts the connection establishment as the active partner. The other communications partner reacts by starting its connection establishment as the passive partner. If both communication partners have initiated their connection establishment, the communication connection is fully established.

Connection parameter assignment

You can assign parameters to establish the connection using a connection description DB with the TCON_Param structure as follows:

- Manually create, assign parameters and write directly to the instruction.
- Supported by connection parameter assignment.

Connection parameter assignment supports the establishment of the connection and should, therefore, be given preference over the other methods.

You specify the following in the connection parameter assignment:

- Connection partner
- Connection type
- Connection ID
- Connection description DB
- Address details according to selected connection type

In addition, you specify here which communication partner activates the connection establishment and which partner establishes a connection passively in response to a request from its communications partner.

See also

Principle of operation of connection-oriented protocols (Page 401)

Overview of connection parameter assignment

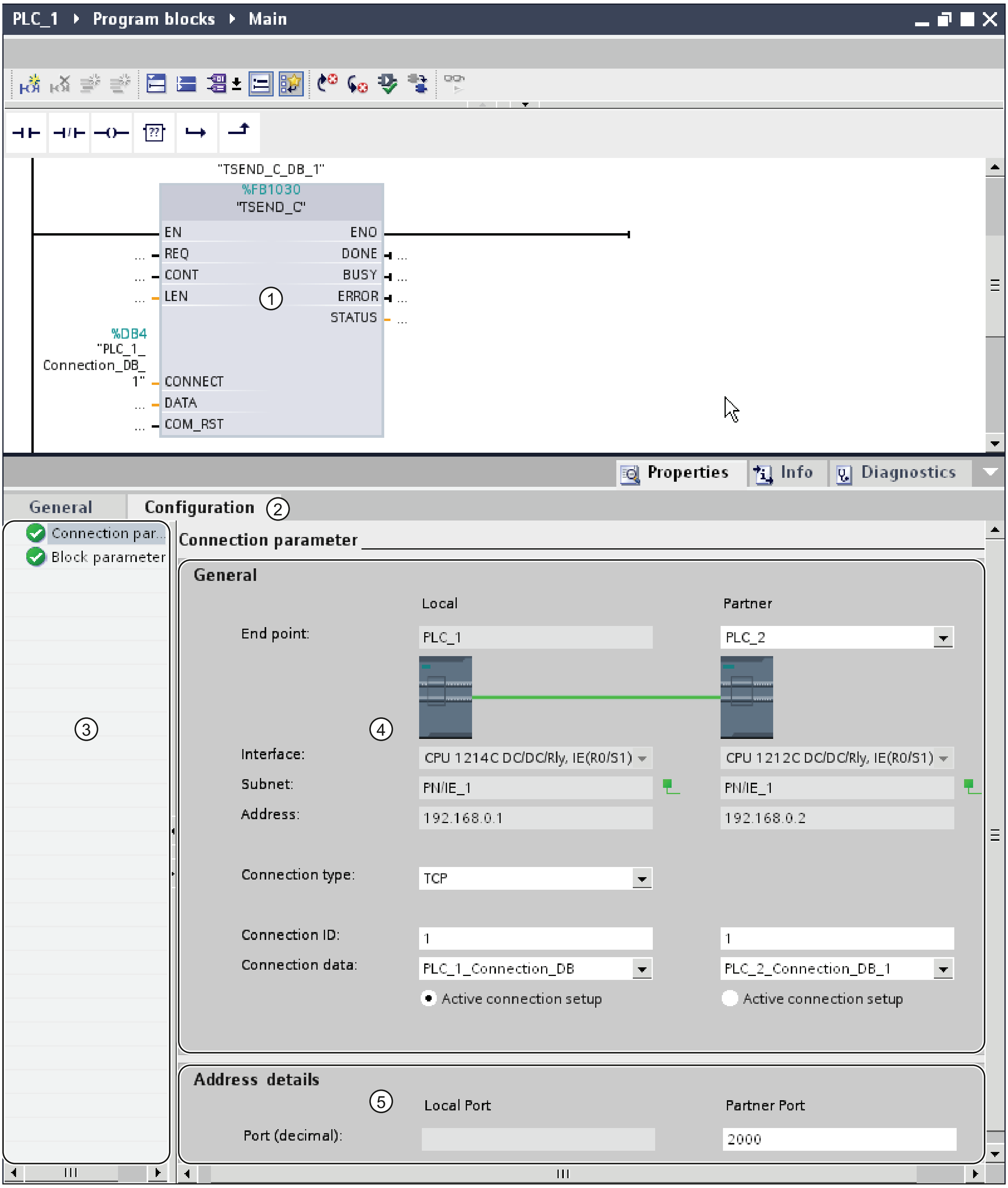
Introduction

You will find the connection configuration in the Inspector window of the program editor if you want to program Open User Communication with the communication instructions TSEND_C, TRCV_C or TCON.

Connection parameter assignment supports the flexible functionality of communications programming: The parameters entered for the connection configuration are stored in an automatically created global DB derived from the structure of the TCON_Param type. You can modify the connection parameters in this connection description DB.

Structure of the connection parameter assignment

The connection parameter assignment is made up of the following components:



Connection parameter

General

	Local	Partner
End point:	PLC_1	PLC_2
Interface:	CPU 1214C DC/DC/Rly, IE(R0/S1)	CPU 1212C DC/DC/Rly, IE(R0/S1)
Subnet:	PN/IE_1	PN/IE_1
Address:	192.168.0.1	192.168.0.2
Connection type:	TCP	
Connection ID:	1	1
Connection data:	PLC_1_Connection_DB	PLC_2_Connection_DB_1
	● Active connection setup	○ Active connection setup

Address details

	Local Port	Partner Port
Port (decimal):		2000

① Communication instruction for TCON, TSEND_C or TRCV_C
② "Configuration" tab in the "Properties" tab
③ Area navigation of the "Configuration" tab
④ General properties of the connection parameters
⑤ Address details of the connection parameters (for selected connection DBs)

"Configuration" tab

Enter the desired connection parameters in the "Configuration" tab. The area navigation of the "Configuration" tab includes the "Connection parameters" group. This group contains the connection parameter assignment. Here, you can enter the parameters for the connections and the address details with system support. Here, you also connect the CONNECT (TCON, TSEND_C, TRCV_C) or ID (TCON, TSEND, TRCV, TUSEND, TURCV) block parameters of the selected communication instructions.

When all the required parameters are assigned, a check mark is set in front of the "Connection parameters" group in the area navigation.

NOTICE
The connection configuration does not check whether the connection IDs and port numbers (TCP, UDP) or TSAPs (ISO-on-TCP) are unique. When you configure Open User Communication, you should, therefore, make sure that the parameter settings are unique within a device.

See also

Parameters of communication connections (Page 403)

Description of the connection parameters

Overview

The following table shows the general connection parameters:

Parameter	Description
End point	The names of the local end point and the partner end point are shown. The local end point is the CPU for which TCON, TSEND_C or TRCV_C is programmed. The local end point is, therefore, always known. The partner end point is selected from the drop-down list. The drop-down list shows all available possible connection partners including unspecified connection partners for devices whose data is unknown in the project. As long as no connection partner is set, all other parameters in the mask are disabled.
Interface	The interface of the local end point is displayed. The partner interface is displayed only after a specified partner end point.
Subnet	The subnet of the local end point is displayed, provided this exists. The partner subnet is displayed only after the partner end point has been selected. If at least one of the two connection partners is not connected with a subnet, the two connection partners are connected with each other. A connection between partners in different subnets is only possible with IP routing. The routing settings can be edited in the relevant interface properties.
Address	The IP address of the local end point is displayed. The IP address of the partner is displayed only after the partner end point has been selected. If you have selected an unspecified connection partner, the input box is empty and has a red background. In this case, you will need to specify a valid IP address.
Connection type	Select the connection type you want to use from the "Connection type" drop-down list: • TCP • ISO-on-TCP • UDP The parameters for the required connection data change depending on the selected connection type.
Connection ID	Enter the connection ID in the input box. You can change the connection ID in the input boxes or enter it directly in TCON. Ensure that the connection ID assigned is unique within the device.
Connection data	The names of the connection description DBs for the connection description structured according to TCON_Param are displayed in the drop-down lists. When you create the connection, one data block each is generated for the specified connection partner and automatically filled with the values from the connection parameter assignment. For the local connection partner, the name of the selected data block is entered automatically in the block parameter CONNECT of the selected TSEND_C, TRCV_C or TCON instruction. For the second connection partner, the connection description DB generated by the first connection partner can also be used directly at the CONNECT input of the TSEND_C, TRCV_C or TCON instructions. With this procedure, you can use the existing connection description DB after selecting the first connection partner or create a new connection description DB. From the drop-down list, you can also reference another valid data block. If a DB is referenced using the CONNECT input parameter of the TSEND_C, TRCV_C or TCON extended instruction and this does not correspond to the structure of a TCON_Param, the drop-down list is shown with no content on a red background.

Parameter	Description
Active connection establishment	Use the "Active connection establishment" check box to specify the active partner of the Open User Communication (only with TCP and ISO-on-TCP).
Port (only with TCP and UDP)	Address component for a TCP or UDP connection. The default after creating a new TCP connection is 2000. You can change the port numbers. The port numbers must be unique on the device!
TSAP (ISO-on-TCP only)	Address component for an ISO-on-TCP connection. The default after creating a new ISO-on-TCP connection is E0.01.49.53.4F.6F.6E.54.43.50.2D.31. You can enter the TSAP-ID with an extension or as an ASCII TSAP. The TSAPs must be unique on the device!

See also

Parameters of communication connections (Page 403)

Assignment of port numbers (Page 406)

TSAP structure (Page 408)

Examples of TSAP assignment (Page 411)

Ability to read back connection description parameters (Page 407)

Creating and assigning parameters to connections (Page 397)

Starting connection parameter assignment

The connection parameter assignment for Open User Communication is enabled as soon as a TCON, TSEND_C or TRCV_C instruction for communication is selected in a program block.

Requirements

- Your project must contain at least one S7-CPU.
- The program editor is open.
- A network is available.

Procedure

To insert the extended instructions for Open User Communication, proceed as follows:

1. Open the task card, pane and folder "Instructions > Communication > Open User Communication".
2. Drag one of the following instructions to a network:
   - TSEND_C
   - TRCV_C
   - TCON

   The "Call options" dialog opens.
3. Edit the properties of the instance DB in the "Call properties" dialog. You have the following options:
   - Change the default name.
   - Select the "Manual" check box to assign your own number.
4. Click "OK" to complete your entry.

Result

A corresponding instance DB is created for the inserted instruction TSEND_C, TRCV_C or TCON.

With TSEND_C, TRCV_C or TCON selected, you will see the "Configuration" tab under "Properties" in the Inspector window. The "Connection parameters" group in area navigation contains the connection parameter assignment that you can now make.

See also

Creating and assigning parameters to connections (Page 397)

Creating and assigning parameters to connections

In the connection configuration for Open User Communication, you can create and configure connections of the TCP, UDP or ISO-on-TCP type.

Requirement

A CPU exists with a TCON, TSEND_C or TRCV_C communication instruction.

Procedure

To create a connection for Open User Communication, proceed as follows:

1. Select a TCON, TSEND_C or TRCV_C block of Open User Communication in the program editor.
2. Open the "Properties > Configuration" tab in the inspector window.
3. Select the "Connection parameters" group. Until you select a connection partner, only the empty drop-down list for the partner end point is enabled. All other input options are disabled.

   The connection parameters already known are displayed:

   – Name of the local end point

   – Interface of the local end point

   – IP address of the local end point
4. In the drop-down list box of the partner end point, select a connection partner. You can select an unspecified device or a CPU in the project as the communications partner. Certain connection parameters are then entered automatically.

   The following parameters are set:

   – Interface of the partner end point

   – Interface of the partner end point

   – IP address of the partner end point

   If the connection partners are networked, the name of the subnet is displayed.
5. In the "Connection data" drop-down list, select the existing connection description DBs or create new connection description DBs. You can subsequently select different connection description DBs in the relevant "Connection data" drop-down lists or change the name of the connection description DBs to create new data blocks:

   – You can also see the selected data block at the interconnection of the CONNECT input parameter of the selected TCON, TSEND_C or TRCV_C instruction.

   – If you have already specified a connection description DB for the connection partner using the CONNECT parameter of the TCON, TSEND_C or TRCV_C instruction, you can either use this DB or create a new DB.

   – If you edit the name of the displayed data block in the drop-down list, a new data block with the changed name but with the same structure and content is generated and used for the connection.

   – Changed names of a data block must be unique in the context of the communications partner.

   – A connection description DB must have the TCON_Param structure.

   – A data block cannot be selected for an unspecified partner.

Additional values are determined and entered after the selection or creation of connection description DBs. The following is valid for specified connection partners:

- ISO-on-TCP connection type
- Connection ID with default of 1
- Active connection establishment by local partner
- TSAP-ID E0.01.49.53.4F.6F.6E.54.43.50.2D.31

The following is valid for unspecified connection partners:

- TCP connection type
- Partner port 2000

6. Enter a connection ID for the connection partner. No connection ID can be assigned to an unspecified partner.

Note

You must enter a unique value for the connection ID at a known connection partner. The uniqueness of the connection ID will not be checked by the connection parameter settings and there will be no default value entered for the connection ID when you create a new connection.

7. Select the connection type you want from the relevant drop-down list. Default values are set for the address details depending on the connection type. You can choose between the following:
   - TCP (partner port 2000)
   - ISO-on-TCP (TSAP-ID E0.01.49.53.4F.6F.6E.54.43.50.2D.31)
   - UDP (local port 2000)
8. You can edit the input boxes in the address details. Depending on the selected protocol, you can edit the ports (for TCP) or the TSAPs (for ISO-on-TCP).
9. Use the "Active connection establishment" check box to set the connection establishment behavior for TCP and ISO-on-TCP. You can decide which communications partner establishes the connection actively.

Changed values are checked immediately for input errors by the connection parameter assignment and entered in the data block for the connection description.

Note

Open User Communication between two communications partners can only work when the program section for the partner end point has been downloaded to the hardware. To achieve fully functional communication, make sure that you load not only the connection description of the local CPU on the device but also that of the partner CPU as well.

See also

Description of the connection parameters (Page 394)

Starting connection parameter assignment (Page 396)

TSAP structure (Page 408)

Assignment of port numbers (Page 406)

Parameters of communication connections (Page 403)

Deleting connections

Introduction

The data of a created connection for Open User Communication is stored in a connection description DB. You can delete the connection by deleting the data block containing the connection description.

Requirement

You have created an Open User Communication connection.

Procedure

To delete a connection, proceed as follows:

1. Select a communications partner for Open User Communication in the project tree.
2. Open the "Program blocks" folder below the selected communications partner.
3. Select the "Delete" command from the shortcut menu of the data block with the connection parameter assignment.

Note

If you are not certain which block to delete, open the extended instruction TCON, TSEND_C or TRCV_C. You will find the name of the data block as the CONNECT input parameter or in the connection parameter assignment as the "Connection data" parameter.

If you only delete the instance DBs of the extended instructions TCON, TSEND_C or TRCV_C, the assigned connections are not deleted as well.

Note

If the connection DB is used by other blocks of the extended instructions, then the corresponding calls, their instance DBs and the combination blocks TSEND_C and TRCV_C are also deleted from the block folder, provided they are not used elsewhere.

This action prevents the program from being inconsistent.

Result

You have deleted the connection.

Note

Insert an extended instruction TCON, TSEND_C or TRCV_C again to reference an existing connection description with the TCON_Param structure via the "Connection data" parameter.

How protocols work

Principle of operation of connection-oriented protocols

Introduction

Connection-oriented protocols establish a logical connection to the communication partner before data transmission is started. After the data transmission is complete, they then terminate the connection, if necessary. In particular, connection-oriented protocols are used for data transmission when reliable, guaranteed delivery is of importance. Several logical connections can exist over one physical line.

Open User Communication supports the following connection types:

- TCP
- ISO-on-TCP
- UDP

For communication partners that do not support an ISO-on-TCP connection, a TCP connection should be used. For these types of communication partners, such as third-party devices or PCs, enter "unspecified" for the partner end point during connection parameter assignment.

Characteristics of TCP

During data transmission via a TCP connection, no information about the length or about the start and end of a message is transmitted. This does not pose a problem during sending because the sender knows the amount of data to be transmitted. However, the receiver has no means of recognizing where one message in the data stream ends and the next one begins. It is therefore recommended that the number bytes to be received (parameter LEN, instruction TRCV/TRCV_C) be assigned the same value as the number of bytes to be sent (parameter LEN, instruction TSEND/TSEND_C).

If the length of the sent data and the length of the expected data do not match, the following will occur:

- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) greater than length of data to be sent (parameter LEN, instruction TSEND/TSEND_C):

  TRCV/TRCV_C copies the received data to the specified receive area (parameter DATA) only after the assigned length is reached. When the assigned length is reached, data of the subsequent job are already being received. As a result, the receive area contains data from two different send jobs. If you do not know the exact length of the first message, you are unable to recognize the end of the first message and the start of the second message.

- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) less than length of data to be sent (parameter LEN, instruction TSEND/TSEND_C):

  TRCV/TRCV_C copies the number of bytes you specified in the LEN parameter to the receive data area (parameter DATA). Then, it sets the NDR status parameter to TRUE (job completed successfully) and assigns RCVD_LEN (amount of data actually received) the value of LEN. With each subsequent call, you receive a further block of the sent data.

Characteristics of ISO-on-TCP

During data transmission via an ISO-on-TCP connection, information regarding the length and the end of a message is also supplied.

If the length of the sent data and the length of the expected data do not match, the following will occur:

- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) greater than length of data to be sent (parameter LEN, instruction TSEND/TSEND_C):

  TRCV/TRCV_C copies all the sent data to the receive data area (parameter DATA). Then, it sets the NDR status parameter to TRUE (job completed successfully) and assigns RCVD_LEN (amount of data actually received) the length of the data sent.

- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) less than length of data to be sent (parameter LEN, instruction TSEND/TSEND_C):

  TRCV/TRCV_C does not copy any data to the receive data area (parameter DATA), but instead supplies the following error information: ERROR=1, STATUS=W#16#8088 (destination buffer too small).

Characteristics of UDP

Information on the length and the end of a message is also supplied during data transmission via a UDP connection.

If the length of the sent data and the length of the expected data do not match, the following will occur:

- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) greater than length of data to be sent (parameter LEN, instruction TUSEND/TSEND_C):

  TURCV/TRCV_C copies all the sent data to the receive data area (DATA parameter). Then, it sets the NDR status parameter to TRUE (job completed successfully) and assigns RCVD_LEN (amount of data actually received) the length of the data sent.

- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) less than length of data to be sent (parameter LEN, instruction TUSEND/TSEND_C):

  TRCV/TRCV_C copies as much data to the receive data area (parameter DATA) as the LEN parameter requests. No further error message is generated. In this case, the user has to call a T_URCV again in order to receive the remaining bytes.

See also

Basics of Open User Communication (Page 391)

Parameters of communication connections

Data block for connection description

A connection description DB with a structure according to TCON_Param is used to configure the communication connections for TCP, UDP and ISO-on-TCP. The fixed data structure of the TCON_Param contains all the parameters that are needed to establish the connection. The connection description DB is automatically created for a new connection by the connection parameter assignment for Open User Communication when the TSEND_C, TRCV_C or TCON instruction is used.

The CONNECT connection parameter of the instance DBs for TSEND_C, TRCV_C or TCON contains a reference to the utilized data block.

Structure of the connection description according to TCON_Param

Byte	Parameter	Data type	Initial value	Description
0 … 1	block_length	UINT	64	Length: 64 bytes (fixed)
2 … 3	id	CONN_OUC	1	Reference to this connection (range of values: 1 to 4095). You specify the value of this parameter for the TSEND_C, TRCV_C or TCON instruction under ID.
4	connection_type	USINT	17	Connection type: • 17: TCP • 18: ISO-on-TCP • 19: UDP
5	active_est	BOOL	TRUE	Identifier for the type of connection establishment: FALSE always applies to UDP, since data can be sent and received via local ID. The following is valid for TCP and ISO-on-TCP: • FALSE: Passive connection establishment • TRUE: Active connection establishment
6	local_device_id	USINT	1	ID for the local PN/IE interface.
7	local_tsap_id_len	USINT	0	Length of parameter local_tsap_id used, in bytes; possible values: • 0 or 2, if connection type = 17 (TCP) Only the value 0 is permissible for the active side. • 2 to 16, if connection type = 18 (ISO-on-TCP) • 2, if connection type = 19 (UDP)
8	rem_subnet_id_len	USINT	0	This parameter is not used.
9	rem_staddr_len	USINT	4	Length of address of partner end point, in bytes: • 0: unspecified, i.e., parameter rem_staddr is irrelevant. • 4: valid IP address in the parameter Parameter rem_staddr (TCP and ISO-on-TCP only)
10	rem_tsap_id_len	USINT	2	Length of parameter rem_tsap_id used, in bytes; possible values: • 0 or 2, if connection type = 17 (TCP) Only the value 0 is permissible for the passive side. • 2 to 16, if connection type = 18 (ISO-on-TCP) • 0, if connection type = 19 (UDP)
11	next_staddr_len	USINT	0	This parameter is not used.

Byte	Parameter	Data type	Initial value	Description
12 … 27	local_tsap_id	ARRAY [1..16] of BYTE	-	Local address component of connection: • TCP and UDP: local port no. (possible values: 1 to 49151; recommended values: 2000...5000); local_tsap_id[1] = high byte of port no. in hexadecimal notation; local_tsap_id[2] = low byte of port no. in hexadecimal notation; local_tsap_id[3-16] = irrelevant • ISO-on-TCP: local TSAP-ID: local_tsap_id[1] = B#16#E0; local_tsap_id[2] = rack and slot of local end points (bits 0 to 4: Slot number, bits 5 to 7: rack number); local_tsap_id[3-16] = TSAP extension, optional Note: Make sure that every value of local_tsap_id is unique within the CPU.
28 … 33	rem_subnet_id	ARRAY [1..6] of USINT	-	This parameter is not used.
34 … 39	rem_staddr	ARRAY [1..6] of USINT	-	TCP and ISO-on-TCP only: IP address of the partner end point, for example for 192.168.002.003: • rem_staddr[1] = 192 • rem_staddr[2] = 168 • rem_staddr[3] = 002 • rem_staddr[4] = 003 • rem_staddr[5-6]= irrelevant
40 … 55	rem_tsap_id	ARRAY [1..16] of BYTE	-	Partner address component of connection • TCP: partner port no. (possible values: 1 to 49151; recommended values: 2000...5000); rem_tsap_id[1] = high byte of port no. in hexadecimal notation; rem_tsap_id[2] = low byte of port no. in hexadecimal notation; rem_tsap_id[3-16] = irrelevant • ISO-on-TCP: partner TSAP-ID: rem_tsap_id[1] = B#16#E0; rem_tsap_id[2] = rack and slot of partner end point (bits 0 to 4: Slot number, bits 5 to 7: rack number); rem_tsap_id[3-16] = TSAP extension, optional • UDP: This parameter is not used.

Byte	Parameter	Data type	Initial value	Description
56 … 61	next_staddr	ARRAY [1..6] of BYTE	-	This parameter is not used.
62 … 63	spare	WORD	W#16#0000	Reserved.

See also

Principle of operation of connection-oriented protocols (Page 401)

Description of the connection parameters (Page 394)

Ability to read back connection description parameters (Page 407)

Overview of connection parameter assignment (Page 392)

TSAP structure (Page 408)

Assignment of port numbers (Page 406)

Assignment of port numbers

Introduction

When an Open User Communication is created, the value 2000 is automatically assigned as the port number.

Permissible values for port numbers are 1 to 49151. You can assign any port number within this range. However, because some ports may already be used depending on the system, port numbers within the range from 2000 to 5000 are recommended.

Overview of port numbers

The following table summarizes the system reactions to various port numbers.

Port no.	Description	System reaction
2000 … 5000	Recommended range	No warning, no error message on entry Port number is permitted and accepted
1 … 1999, 5001 … 49151	Can be used, but is outside the recommended range	Warning message on entry Port number is permitted and accepted
20, 21, 25, 80, 102, 135, 161, 34962 … 34964	Can be used conditionally*	
53, 80, 102, 135, 161, 162, 443, 520, 9001, 34962 … 34964	Can be used conditionally**	

* These ports are used by TSEND_C and TRCV_C with the TCP and UDP connection types.

** These ports are blocked depending on the function scope of the utilized S7-1200 CPU. The documentation of the respective CPUs provides the assignment of these ports.

See also

Description of the connection parameters (Page 394)

Creating and assigning parameters to connections (Page 397)

Ability to read back connection description parameters

Changing parameter values in the connection description

The connection description for exactly one connection of the Open User Communication is entered from the connection parameter assignment in the connection description DB.

You can change the parameter values of the connection description DB outside of the connection parameter assignment in the user program. The structure of the connection description cannot be changed.

Connection description DBs containing values you changed subsequently can be read back from the connection parameter assignment. Under "Properties > Configuration > Connection parameters", the Inspector window displays only the connection parameters stored in the connection description DB.

The connection parameter assignment does not support nested entries of connection descriptions in DB types that can only be found via offset referencing (for example, Global-DB).

Ability to read back individual connection parameters

For the "Address" parameter of the communication partner in a TCP or ISO-on-TCP connection, its IP address is displayed from the "rem_staddr" parameter of the connection description.

The following values can also be reloaded from the connection description:

- Connection type
- Local connection ID:
- Active/passive connection establishment (only with UDP)
- Local TSAP (ISO-on-TCP only)
- Partner TSAP (ISO-on-TCP only)
- Local port (only with TCP and UDP)
- Partner port (TCP only)

The values of the connection ID parameters of the communication partner, the connection data, as well as the connection establishment, are not included in the connection description in the local connection description DB. Consequently, these parameters cannot be displayed when the connection parameter assignment is reopened. The connection establishment of the partner results from the local connection establishment and is therefore also displayed.

A new communication partner can be selected at any time in the "Partners" drop-down list box.

When a CPU recognized in the project is selected as a specified communication partner, the entry options for the connection ID and the connection data are shown again.

See also

Parameters of communication connections (Page 403)

Description of the connection parameters (Page 394)

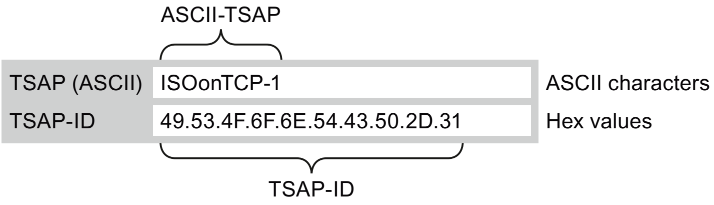
TSAP structure

Introduction

For an ISO-on-TCP connection, Transport Service Access Points (TSAPs) must be assigned for both communication partners. TSAP IDs are assigned automatically after an ISO-on-TCP connection is created. To ensure the uniqueness of TSAP IDs within a device, you can change the preassigned TSAPs in the connection parameter assignment.

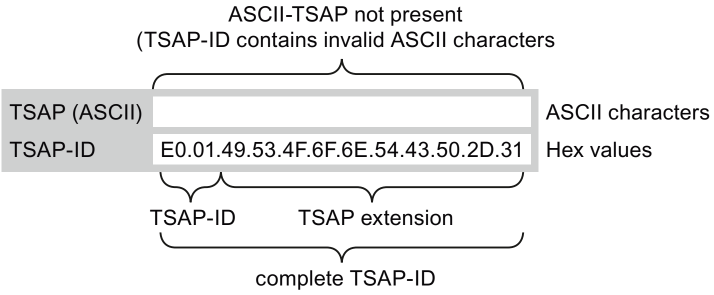
Structure of TSAPs

You must comply with certain rules when assigning TSAPs. A TSAP must contain a certain number of bytes, which are able to be displayed and entered as hexadecimal values (TSAP-ID) or as ASCII characters (ASCII-TSAP):



Entries or changes of the TSAP-ID or the ASCII-TSAP in the corresponding entry fields always take effect in the other display format as well.

If a TSAP contains no valid ASCII characters, the TSAP is displayed only as TSAP-ID and not as ASCII-TSAP. This is the case after a connection is created. The first two hex characters as TSAP-ID identify the communication type and the rack/slot. Because these characters are not valid ASCII characters for a CPU, the ASCII-TSAP is not displayed in this case.



In addition to the rules for length and structure of TSAPs, you must also ensure the uniqueness of the TSAP-ID. The assigned TSAPs are not automatically unique.

Length and content of TSAPs

A TSAP is structured as follows:

- TSAP-ID with TSAP extension

  Length = 2 to 16 bytes

  x_tsap_id[0] = 0xE0 (Open User Communication)

  x_tsap_id[1] (bits 0 to 4) = slot number of CPU

  x_tsap_id[1] (bits 5 to 7) = rack number of CPU

  x_tsap_id[2...15] = any characters (TSAP extension, optional)

  (x = loc (local) or x = rem (partner))

- TSAP-ID as ASCII-TSAP

  Length = 3 to 16 bytes

  x_tsap_id[0 to 2] = 3 ASCII characters (0x20 to 0x7E)

  x_tsap_id[3...15] = any characters (optional)

  (x = loc (local) or x = rem (partner))

The following table shows the schematic structure of a TSAP-ID:

TSAP-ID	tsap_id_len	tsap_id[0]	tsap_id[1]	tsap_id[2..15]	tsap_id[3..15]
...with extension	2...16 bytes	0xE0	0x01 (0x00)*	Extension (optional)	Extension (optional)
...as ASCII-TSAP	3...16 bytes	0x20...0x7E	0x20...0x7E	0x20...0x7	Any (optional)

*A recognized CPU is normally inserted on rack 0 in slot 1. For this reason, hex value 01 is valid for the second place of the TSAP-ID with extension. If the connection partner is an unspecified CPU, for example, a third-party device, the hex value 00 is also permissible for the slot address.

Note

For unspecified communication partners, the local TSAP-ID and the partner TSAP-ID can have a length of 0 to 16 bytes, in which all hex values from 00 to FF are permitted.

ASCII code table for entry of ASCII TSAPs

For entry of an ASCII-TSAP in the connection parameter assignment, only hexadecimal values from 20 to 7E are permitted:

Code	..0	..1	..2	..3	..4	..5	..6	..7	..8	..9	..A	..B	..C	..D	..E	..F
2..		!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
3..	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4..	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5..	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6..	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7..	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

See also

Examples of TSAP assignment (Page 411)

Description of the connection parameters (Page 394)

Creating and assigning parameters to connections (Page 397)

Examples of TSAP assignment

The following examples illustrate various aspects of editing TSAPs:

- Example 1: Creating a new connection for PLC-PLC communication
- Example 2: Entry of a local ASCII-TSAP
- Example 3: Entry of a TSAP extension in the TSAP-ID
- Example 4: Incorrect editing of the TSAP-ID
- Example 5: Entry of an ASCII-TSAP via the "TSAP-ID" entry field

Example 1: Creating a new connection for PLC-PLC communication

Once you have created a new connection with two PLCs for the Open User Communication, the TSAP extension "ISOonTCP-1" is assigned automatically.

This TSAP extension produces the TSAP-ID E0.01.49.53.4F.6F.6E.54.43.50.2D.31, which is entered automatically in the connection description DB and in the entry fields of the local and the partner TSAP. The entry fields of the ASCII-TSAPs remain empty:

	Local TSAP	Partner TSAP
TSAP (ASCII)		
TSAP-ID	E0.01.49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

You can change the values in the entry fields of the TSAP-ID and the ASCII-TSAP at any time.

The entry field of the TSAP-ID shows the complete TSAP stored in the data block of the connection description. The TSAP-ID with TSAP extension, which is limited to 16 characters, is not displayed in the "TSAP (ASCII)" entry field because the character E0 does not represent a valid character for the ASCII-TSAP.

If the displayed TSAP-ID is a valid ASCII-TSAP, it is displayed in the "TSAP (ASCII)" entry field.

Changes in the entry fields for TSAP-ID and ASCII-TSAP affect the other field.

Example 2: Entry of a local ASCII-TSAP

If you have created a new connection and assigned an ASCII value for the local TSAP in the "TSAP (ASCII)" entry field, for example, "ISOonTCP-1", the resulting TSAP-ID is created automatically.

When you exit the "TSAP (ASCII)" entry field, the number of ASCII characters is checked automatically for compliance with the limit (3 to 16 characters) and the resulting TSAP-ID is entered into the corresponding entry field:

	Local TSAP	Partner TSAP
TSAP (ASCII)	ISOonTCP-1	
TSAP-ID	49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

Example 3: Entry of a TSAP extension in the TSAP-ID

If, following creation of a connection and entry of an ASCII-TSAP (see examples 1 and 2) in the entry field of the local TSAP-ID, you add the prefix "E0.01" to the TSAP value, the ASCII-TSAP will no longer be displayed when the entry field is exited.

	Local TSAP	Partner TSAP
TSAP (ASCII)		
TSAP-ID	E0.01.49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

Once you have exited the entry field of the TSAP-ID, a check is performed automatically to determine whether the first character of the TSAP-ID is a valid ASCII character. Since the character "E0" now present in the TSAP-ID is not a valid character for the ASCII-TSAP, the "TSAP (ASCII)" entry field no longer displays an ASCII-TSAP.

If a valid ASCII character is used, the check for compliance with the length specification of 2 to 16 characters follows.

Example 4: Incorrect editing of the TSAP-ID

If you remove the hex value "E0" from a TSAP-ID beginning with "E0.01", the TSAP-ID now begins with "01" and therefore no longer complies with the rules and is invalid:

	Local TSAP	Partner TSAP
TSAP (ASCII)		
TSAP-ID	01.49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

After the entry field is exited, a message is output because the TSAP-ID is neither a valid ASCII-TSAP (this would have to have a hex value in the range from 20 to 7E as the first value) or a valid TSAP-ID (this would have to have the identifier "E0" as the first value).

Example 5: Entry of an ASCII-TSAP via the "TSAP-ID" entry field

If you remove the value "01" in addition to the value "E0" from the incorrect TSAP-ID in example 4, the TSAP-ID begins with the hex value 49. This value is within the permissible range for ASCII-TSAPs:

	Local TSAP	Partner TSAP
TSAP (ASCII)		
TSAP-ID	49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

When you exit the entry field, the TSAP-ID is recognized as a valid ASCII-TSAP and the resulting ASCII-TSAP "ISOonTCP-1" is written to the "TSAP (ASCII)" entry field.

See also

TSAP structure (Page 408)

Description of the connection parameters (Page 394)

8.1.3.3 Displaying and configuring topology

Overview of the topology view

Functions of the topology view

The topology view is one of three working areas of the hardware and network editor. You undertake the following tasks here:

- Displaying the Ethernet topology
  - Displaying all the PROFINET devices and passive Ethernet components of the project along with their ports
  - Displaying interconnections between the ports
  - Displaying corresponding logical networks
  - Display diagnostic information of all ports
- Configuring the Ethernet topology
  - Creating, modifying and deleting interconnections of the ports
  - Renaming stations, devices, interfaces or ports
  - Adding PROFINET devices and passive Ethernet components to the project from the hardware catalog
- Identifying and minimizing differences between the desired and actual topology
  - Running an offline/online comparison of Ethernet modules, ports and port interconnections
  - Adopting existing online topology information in the offline project

Differences between network view and topology view

- The network view shows all the logical subnets of the project.

  The topology view shows all Ethernet components of the project. These include passive components such as switches, media converters and cables.

  Note

  Stations with non-Ethernet components are also displayed if the station has a least one Ethernet component.

- The position of a device in the network view and its position in the topology view are not dependent on each other; in other words, the same device generally appears at different locations in the two views.
- If you open the hardware catalog in the topology view, you only see devices with an Ethernet interface.

Structure of the topology view

The topology view (Page 313) essentially consists of a graphic area (called the graphic view below) and a table area (called the table view below).

Which functions are there in the graphic view and which functions are there in the table view?

- Displaying the Ethernet topology

Function	Graphic view	Table view
Displaying all the PROFINET devices and passive Ethernet components of the project along with their ports	yes	yes
Display interconnections between the ports (including type of medium)	yes	yes
Displaying corresponding logical networks	no	yes
Displaying properties of the cables between the ports	no	yes
Display diagnostic information of all ports	yes	yes

- Configuring the Ethernet topology

Function	Graphic view	Table view
Creating, modifying and deleting interconnections of the ports	• Create: yes • Modify: no • Delete: yes	• Create: yes • Modify: yes • Delete: yes
Renaming stations, devices, interfaces or ports	no	yes
Adding PROFINET devices and passive Ethernet components to the project from the hardware catalog	yes	no

- Identifying and minimizing differences between the desired and actual topology

Function	Graphic view	Table view
Running an offline/online comparison of Ethernet modules, ports and port interconnections	no	yes
Adopting existing online topology information in the offline project	no	yes

Starting the topology view

Requirement

The device or network view is open in the hardware and network editor.

Procedure

To start the topology view of your project, follow these steps:

1. Click on the "Topology view" tab.

Or:

1. Open the network view of the hardware editor.
2. Select a PROFINET device or a PROFINET module.
3. Select the "Go to topology view" command in the shortcut menu.

Result

The graphic view of the topology view is started. If you opened the topology view using the shortcut menu, the selected component remains selected after the change of view.

Displaying topology

Displaying the graphic view of the configured topology

What is shown?

The graphic view of the configured topology shows the following:

- Configured PROFINET devices and passive Ethernet components along with their ports
- Configured stations with non-Ethernet components if there is at least one Ethernet component in the station
- Configured interconnections between the ports

Type of display

The way in which the graphic view of the topology view and the network view are displayed is very similar:

- Compared with the device view, components are shown in a simplified form.
- The interconnections between ports are shown as horizontal and vertical lines. These are dashed when an interconnection between a tool changer port and its possible partner ports is involved.

Displaying the table view of the configured topology

What is shown?

The table view of the configured topology shows exactly the same content as the graphic view except for the logical PROFINET subnets.

- All the configured PROFINET devices and passive Ethernet components along with their ports
- All the configured stations with non-Ethernet components if there is at least one Ethernet component in the station
- Configured interconnections between the ports

  For each port with the "Alternative partner port" property, there are as many completed rows as there are potential partner ports plus one empty row.

Type of display

As the name implies, the table view of the topology view consists of a table, the topology overview table. It is structured like the network overview table. It consists of the following columns:

- Device / port

  This is the most important column of the table. The entries in this column have a hierarchical structure with the PROFINET ports being the last element in the hierarchy. You can expand and collapse the hierarchical entries. For a CPU, for example, an entry consists of the following elements:

  – Station name

  – Device name

  – Name of the PROFINET interface

  – Names of the ports

  Note: All the other columns only have entries in the rows containing the port names.
- Type (as default, this column is not displayed)

  Shows what type of station, device or interface the table row relates to or whether it belongs to a port.
- Order no. (as default, this column is not displayed)

  Order no. of device
- Subnet (as default, this column is not displayed)

  Configured subnet to which the interface belongs
- Master / IO system (as default, this column is not displayed)

  Shows whether or not the interface belongs to a PROFIBUS DP master system or a PROFINET IO system.
- Device address (as default, this column is not displayed)

  Configured address of the interface in the subnet

- Partner station

  Name of the station that contains the partner port
- Partner device

  Name of the device that contains the partner port
- Partner interface

  Interface to which the partner port belongs
- Partner port
- Cable data

  Contains the cable length and the signal delay of the cable connecting the ports

Basic functions for tables

The topology overview table supports the following basic functions for editing a table:

- Displaying and hiding table columns

  Note: The columns that define the configuration cannot be hidden.
- Optimizing column width
- Displaying the meaning of a column, a row or cell using tooltips.

Displaying the diagnostics status of ports and cables in the graphic view

Requirements

The graphic view of the topology view is open.

Procedure

To determine the diagnostics status of the port, follow these steps:

1. Go online with the required component or components.

Result

The following icons are displayed:

- The corresponding diagnostics icon is displayed for each device.
- If there is an error in at least one lower-level component, the diagnostics icon "Error in lower-level component" is also displayed in the left-hand lower corner of the diagnostics icon.
- The corresponding diagnostics icon is displayed for each port.
- Every cable between two ports that are online has the color that matches its diagnostics status.

You will find the possible diagnostics icons for ports and the color coding of Ethernet cables in the description of hardware diagnostics. See: Determination of online status and display using symbols (Page 649)

Showing the diagnostics status of hardware components in the table view

Requirement

The table view of the topology view is open.

Procedure

To obtain the diagnostics status of hardware components of the topology overview table, follow these steps:

1. Go online with the required components.

Result

The following icons are displayed at the left-hand edge of the topology overview table in each row that belongs to the component involved:

- The diagnostics icon belonging to the hardware component is displayed.
- If the hardware component has lower-level components and if there is an error in at least one of the lower-level components, the diagnostics icon "Error in lower-level component" is also displayed in the left-hand lower corner of the diagnostics icon of the hardware component.

For the possible diagnostics icons for hardware components, refer to the description of hardware diagnostics. See: Determination of online status and display using symbols (Page 649)

Note

The display of the diagnostics status of hardware components in the topology overview table and the network overview table is identical.

Running an offline/online comparison and displaying the results

Requirements

The topology view is open. There may be an online connection to one or more devices, but this is not actually necessary.

Procedure

To find the differences between the configured and the actual topology, follow these steps:

1. Click the "Compare offline/online" button in the toolbar of the topology overview.

Result

The "Partner station", "Partner interface" and "Cable data" columns in the topology overview table are removed.

Two additional groups of columns are added to the right-hand side of the table and these are initially empty:

- On the far right, columns for the topology to be identified online are added.
- Between the columns for the offline and the online topology, the "Status", "Action" and "Description" columns are added to show the result of the offline/online comparison.

> **Note**
>
> As default, the "Description" column is not displayed.

The following buttons are enabled in the toolbar of the table:

Button	Name	Meaning
🔄	Update	The detection of the existing online topology is started again.
⇲	Synchronize	• Adopt the port interconnections identified online in the project (Page 429) • Adopt the devices identified online in the project (Page 429)

After the actual topology has been identified, the added columns are filled. This is discussed in more detail in the following section.

> **Note**
>
> A difference between offline and online view is displayed for that port connected with the PG/PC which is only available online. This is because the PG/PC cannot be configured offline.

Columns for the topology identified online

The following columns are displayed:

- "Device / port"
- "Type" (as default, this column is not displayed)
- "Order no." (as default, this column is not displayed)
- "IP address" (as default, this column is not displayed)
- "Partner device"
- "Partner port"
- "Cable data"

Columns for the result of the offline/online comparison

The following columns are displayed:

- "Status"

  The result of the offline/online comparison is shown here in the form of diagnostics icons. The following icons are possible:

Diagnostics icon	Meaning
❗	Differing topology information in at least one lower-level component
●	Identical topology information
◐	Topology information only available offline or device is disabled.
◑	Topology information only exists online
◖◗	Differing topology information

- "Action"

  The possible actions are shown here in the form of icons. The following icons are possible:

Icon	Meaning
=	No action possible
←	Adopt the interconnection found online

- "Description"

  This column describes the selected action in words.

Configuring topology

Interconnecting ports

Overview

Interconnecting ports in the topology view

In the topology view, you have the following options for interconnecting ports:

- in the graphic view (Page 422)
- in the graphic view of a tool changer (Page 424)
- in the table view (Page 423)
- in the table view of a tool changer (Page 424)
- by adopting port interconnections identified online (Page 429)

What effects does the interconnection of ports have on the network view?

Note

In the properties of a subnet in the network view, you can specify that when a port interconnection is created between two devices that are not networked, this subnet is used.

When you create an interconnection between two ports, the following effects are possible in the network view:

- If the corresponding interfaces are not networked: If you have specified a default subnet, this is used. Otherwise a new subnet is created to connect the two interfaces.
- If one (and only one) of the two interfaces involved is networked: The non-networked interface is connected to the same subnet as the already networked interface.
- In all other cases: The corresponding interfaces are not connected to a logical subnet.

Interconnecting ports in the graphic view

Requirements

You are in the graphic view of the topology view.

Procedure – Creating a new interconnection between two ports

To interconnect a port of a device with a port of another device, follow these steps:

1. Place the mouse cursor on the port you want to interconnect.
2. Click with the left mouse button and hold it down.
3. Move the mouse pointer.

   The pointer now shows the networking symbol to indicate "Interconnect" mode. At the same time, the mouse pointer shows the lock symbol that will only disappear when the pointer is on a valid target.
4. Now drag the mouse cursor to the target port. You can do this while holding down or after releasing the mouse button.
5. Now release the left mouse button or press it again (depending on your previous action).

Result: A new port interconnection is created.

Interconnecting ports in the table view

Which actions are possible with port interconnections in the table view?

The following actions are possible with port interconnections in the table view:

- Create new port interconnection
- Modify existing port interconnection
- Delete existing port interconnection

Requirement

The row with the port whose interconnection you want to create, modify or delete is visible in the topology overview.

Procedure

To create the interconnection of a port for the first time, to modify it or delete it, follow these steps:

1. Move the mouse pointer to the "Partner port" column in the row of the source port.
2. Click on the drop-down list there.
3. Select the required partner port (when creating a new interconnection or changing the port interconnection) or the "Not interconnected" entry (when deleting a port interconnection).

Result

The required action is performed. The new partner port (after creating or modifying a port interconnection) is displayed in the "Partner port" column otherwise the "Select port" entry (after deleting a port interconnection).

Interconnecting a port with more than one partner port in the graphic view

Requirement

- You have assigned parameters to a port of a PROFINET device with the "Alternative partners" property and have specified the possible partner ports.
- The graphic view of the topology view is open.

Procedure

1. Interconnect this port (called source port below) with one of the partner ports you have specified (called target port below).
2. Interconnect the source port with an additional target port.

   You can do this in several ways:

   – Drag the mouse pointer from an already interconnected partner port to a target port.

   – Drag the mouse pointer from an interconnection that has already been created to a target port.

   – Drag the mouse pointer from a target port to an already interconnected partner port.

   – Drag the mouse pointer from a target port to an already created interconnection.
3. If necessary, repeat the step above one or more times.

Result

An interconnection is created between the source port and the alternative partner ports. This is indicated by a dashed line.

Interconnecting a port with more than one partner port in the table view

Which actions are possible with port interconnections to multiple partner ports in the table view?

When working with a tool changer, the following actions can be performed with port interconnections to multiple partner ports in the table view:

- Create new port interconnection
- Modify existing port interconnection
- Delete existing port interconnection

Requirement

- You have assigned parameters to a port of a PROFINET device with the "Alternative partners" property and have specified the possible partner ports.
- The row with the port whose interconnection you want to create, modify or delete is visible in the topology overview.

Procedure

To create the interconnection of a port to one or more partner ports for the first time, to modify it or delete it, follow these steps:

1. Move the mouse pointer to the "Partner port" column in the row of the source port.
2. Click on the drop-down list there.
3. Select the required partner port (when creating a new interconnection or changing the port interconnection) or the "Not interconnected" entry (when deleting a port interconnection).

Result

The required action is performed:

- If you are creating an interconnection, a new row is inserted in the topology overview table. The new partner port is displayed there in the "Partner port" column.
- If you make a change, the new partner port is displayed in the "Partner port" column.
- If you are deleting, the row with the previous port interconnection is deleted.

> **Note**
>
> With a tool changer, there are normally several rows for a port with port interconnections to several partner ports. The last row is always an empty row. The first row can be edited, all other rows are read-only.

Renaming stations, devices, interfaces or ports

Rename a station, a device, an interface or a port

Requirement

The table view of the configured topology is open.

Procedure

To rename a station, a device, an interface or a port, proceed as follows:

1. Click twice in the relevant field of the topology overview table (the second click starts the editing mode).
2. Enter the new name and then press the ENTER key (this closes editing mode).

Result

The object is renamed.

Offline/online comparison

Automatic assignment of devices by offline/online comparison

Overview

During the offline/online comparison, the configured topology is compared with the actual existing topology. Devices identified online are automatically assigned to configured devices as far as this is possible.

Start of availability detection

You start the availability detection the first time by clicking the "Compare offline/online" button in the toolbar of the topology overview.

You restart availability detection by clicking the "Update" button.

Note

The availability detection can take several seconds. During this time, no user input is possible.

Automatic assignment

A device identified online is automatically assigned to a configured device if the following properties of the two devices match up:

- Device name
- Order number
- Number of ports

The following section describes the situations that can occur and what action you can take:

- Identical port interconnections

  This is the ideal situation. No action is necessary here.

"Action" column	Meaning
●○	No action

- There are interconnections for the identified and configured device, there are however differences.

  The following actions are possible:

  – If it is possible to adopt the online configuration

"Action" column	Meaning
◖●	Adopt online interconnection (Page 429)
●●	No action

  – If it is not possible to adopt the configuration

"Action" column	Meaning
●●	No action

- The interconnection only exists online.

  The following actions are possible:

  – If it is possible to adopt the online configuration

"Action" column	Meaning
◐●	Adopt online interconnection (Page 429)
○●	No action

  – If it is not possible to adopt the configuration

"Action" column	Meaning
○●	No action

- The interconnection only exists in the configuration.

  The following actions are possible:

"Action" column	Meaning
◐○	Adopt the online interconnection (Page 429), in other words, the interconnection in the configuration will be deleted
●○	No action

No automatic assignment

In the following situations, no automatic assignment is possible:

- No device can be identified online to match a configured device. In this case the corresponding columns in the "Online topology" area of the topology overview table are empty.

  In this case, you should add the already configured device to your system or delete the configured device from the configuration.

- A device identified online cannot be assigned to any configured device. In this case the corresponding columns in the "Offline topology" area of the topology overview table are empty.

  In this case, you can adopt the device identified online in the project (Page 429).

Adopt the port interconnections identified online in the project

Requirements

You have run an offline/online comparison in the topology view. The result of this is that at least one device identified online was automatically assigned to a configured device, but that there are differences relating to the interconnection.

Procedure

To adopt one more port interconnections identified online in the project manually, follow these steps:

1. Select the value "Adopt" in the "Action" column for a port of a configured device to which a device identified online was assigned.
2. Repeat the step if necessary for other ports of the same configured device.
3. Repeat the steps up to now if necessary for other configured devices to which devices identified online were assigned and for which there are differences relating to the interconnection.
4. Click the "Synchronize" button.

Result

The port interconnections identified online and the cable information for the corresponding devices are adopted in the project. Successful adoption is indicated by the diagnostics icon "Identical topology information" for each port.

Note

If other port interconnections are recognized for a device identified online and these differ from those that exist in the project, adopting these in the project means that the port interconnections that were previously in the project are replaced by those identified online. If no port interconnections are detected for a device identified online, adopting in the project means that all the port interconnections of this device are deleted in the project.

Adopt the devices identified online in the project

Requirements

You have run an offline/online comparison in the topology view. The result of this is that at least one device identified online could not be assigned to any configured device.

Procedure

To adopt one more devices identified online in the project manually, follow these steps:

1. For a configured device without an online partner, move the mouse pointer to the "Device/port" column of the online topology.
2. Select the device you want to assign to the configured device from the drop-down list of this box.
3. Repeat the previous steps if necessary for other configured devices without an online partner.

Result

The selected device that was identified online is moved up from the end of the table. Following this, it is in the row of the configured device to which you have just assigned it.

8.1.4 Creating configurations

8.1.4.1 Configurations for automation systems

Addressing modules

Addressing modules

Introduction

In the device view you see the addresses or address ranges of the modules in the I-address and Q-address columns. There are other addresses as well, which are explained below.

I/O address

I/O addresses are required to read inputs and/or set outputs in the user program.

Input and output addresses are assigned automatically when inserting modules in the rack. The address of the first channel is the start address of a module. The addresses of the other channels are derived from this start address. The address end is derived from the module-specific address length.

Device address (e.g., Ethernet address)

Device addresses are addresses of programmable modules (Industrial Ethernet addresses). They are required to address the different stations of a subnet, e.g. to load a user program into a CPU.

Hardware ID for identification of modules or functional units of modules

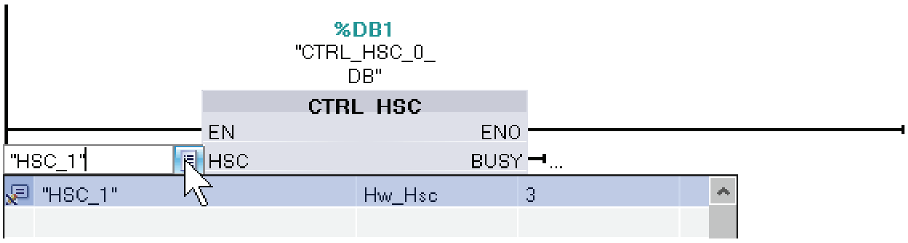
In addition to the I addresses and Q addresses, a hardware identifier is also assigned automatically and is used to identify the module. Functional units of a module, for example an integrated counter, also receive a hardware identifier.

The hardware identifier consists of a whole number and is output by the system along with diagnostics alarms to allow the faulty module or functional unit to be localized.

You also use the hardware identifier for a number of instructions to identify the relevant module on which the instruction will be used.

The hardware ID cannot be changed.

Example: Identifying a high-speed counter



The hardware ID is assigned automatically when components are inserted in the device or network view and in the constants table of the PLC tags. A name is also assigned automatically for the hardware ID. These entries in the constants table of the PLC tags cannot be changed either.

See also

Specifying input and output addresses (Page 431)

Assigning addresses to a location in the program (Page 432)

Introduction to loading a configuration (Page 523)

Specifying input and output addresses

Default input and output addresses are set automatically. You can, however, change the address assignment later.

All addresses of modules are located in the process image area. The process image is automatically updated cyclically.

Requirement

You are in the device view.

Procedure

To change the preset address range proceed as follows:

1. In the device view, click on the module for which you want to set the start address.
2. Go to "I/O addresses" in "Properties" in the inspector window.
3. Under "Start address" enter the required start address.
4. Press <Return> or click on any object to accept a modified value.

If you have entered an invalid address, a message indicating the next available address is displayed.

Note

You can also change the addresses directly in the device overview.

See also

Editing properties and parameters (Page 345)

Assigning addresses to a location in the program

You can assign addresses or symbols from the I/O channels of the modules directly to the locations in the program.

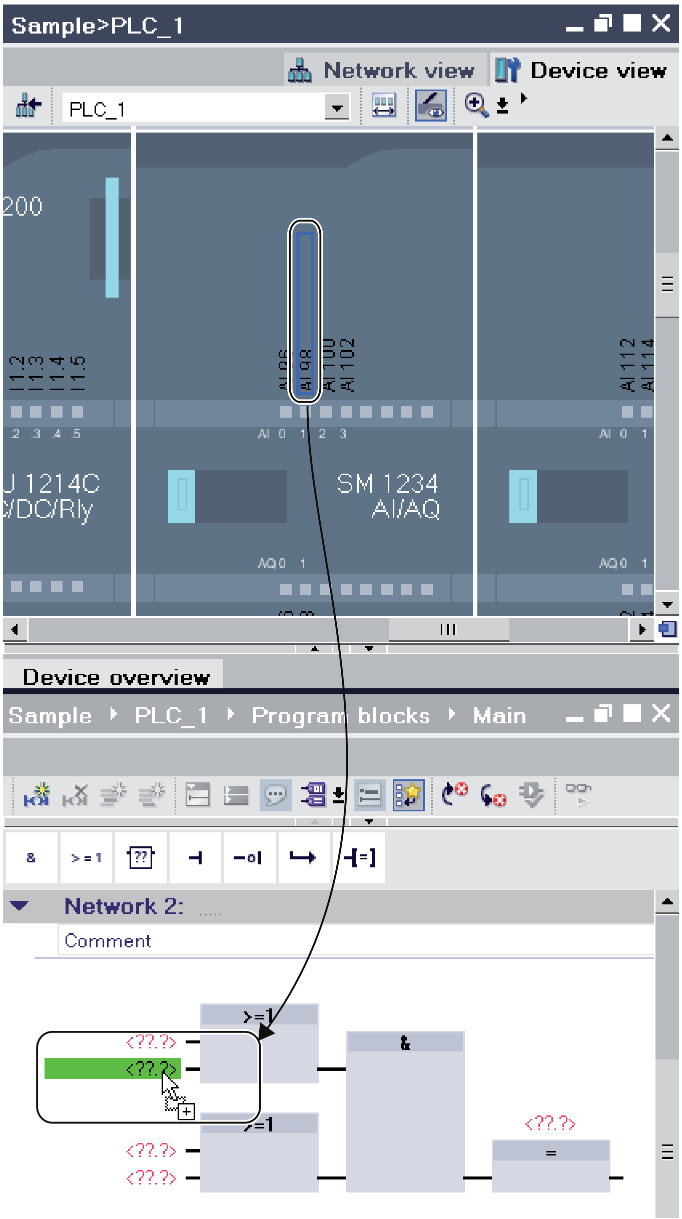
Requirement

The device view of the hardware and network editor as well as the instruction window of the program editor must be opened and arranged one below the other.

Procedure

Proceed as follows to assign addresses of modules and locations in the program:

1. In the device view, navigate to the module with the desired I/O channel.
2. Use the zoom function to specify a magnification of at least 200%: At a magnification level of 200% and higher, the labels of the individual address channels are displayed and can be edited.
3. In the program, navigate to the block with the matching location.
4. Drag the desired address or tag to the appropriate location of the block or the I/O channel of the module.



The module address or tag will be assigned to the location in the program, or the address or tag in the program will be assigned to a module I/O channel.

Signal board

Inserting a signal board in a CPU

Introduction

Signal boards allows you to increase the number of the S7-1200 CPU's own inputs and outputs. Just like all other hardware components, you will find signal boards in the hardware catalog. However, you do not insert signal boards in the rack like other modules but directly in a slot of the CPU itself.

Note the following points when using a signal board:

- Each CPU can have only one signal board inserted in it.
- A signal board can only be inserted when the slot in the CPU is free.

There are various ways of inserting a signal board in a CPU:

- Double click on a signal board in the hardware catalog when there is a free slot in the CPU
- Drag from the hardware catalog to a free slot in the CPU
- Shortcut menu of a signal board in the hardware catalog for copying and pasting

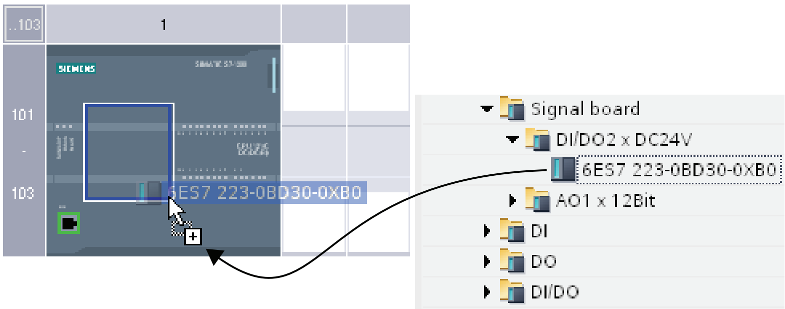
Requirement

- The hardware catalog is open.
- The S7-1200 CPU has a free slot for the signal board.

Inserting a signal board in a CPU

To insert a signal board in a CPU, proceed as follows:

1. Go to the required signal board in the hardware catalog.
2. Select the signal board you want to use.
3. Drag the signal board to the free slot in the CPU.



You have now inserted a signal board in the slot of the CPU.

If you are in the network view, you can also drag a signal board to a device. If the CPU has a an empty slot for a signal board, the signal board is inserted automatically into this slot.

Configurations for Web server

What you need to know about web servers

Introduction

The web server allows you to monitor the CPU via the Internet or the intranet of your company. This permits evaluation and diagnostics over long distances.

Messages and status information are visualized on HTML pages.

Web browser

You need a web browser that supports HTML 1.1 to access the HTML pages of the CPU.

The following web browsers, for example, are suitable for communication with the CPU:

- Internet Explorer (version 6.0 and higher)
- Mozilla Firefox (V1.5 and higher)
- Opera (version 9.0 and higher)
- Netscape Navigator (version 8.1 and higher)

Reading information via the web server

The information listed in the following can be read from the CPU. Availability of the respective web pages depends on the CPU and its firmware version:

Page/information	Description
Intro	Entry page for the standard web pages
Start Page Start page with general CPU information	The start page provides an overview of general information on the CPU, the name of the CPU, the type of CPU and basic information on the current operating state.
Identification Identification information	Displays the static identification information such as serial number, order number and version numbers.
Diagnostic Buffer Diagnostic information	Displays the content of the diagnostics buffer with the most recent entries.
Module Information Module information	Displays whether the centrally inserted components of a station are OK, whether there are maintenance requirements or components cannot be reached, for example.
Communication Communication	Displays the communication connections during open communication (OUC); displays resources and address parameters.
Varable Status Tags	Displays the status of operands of the user program to monitor and change the values.

Page/information	Description
Data Logs	Data logs in CSV format to transfer to the hard disk of the programming device. The data logs are created with data log instructions in the user program and filled with data.
User Pages User pages (if custom web pages have been configured and loaded)	The user web pages provide a list of web pages with customized web applications.

Web access to the CPU via PG/PC

Proceed as follows to access the web server:

1. Connect the client (PG/PC) to the CPU via the PROFINET interface.
2. Open the web browser.

   Enter the IP address of the CPU in the "Address" field of the web browser in the format http://ww.xx.yy.zz (example: http://192.168.3.141).
   The start page of the CPU opens. From the start page you can navigate to further information.

Standard web pages

Requirements for web access

The requirements for access to standard CPU web pages are explained in the following, as well as the effects of missing or existing configuration information.

Requirements

The web server must be started.

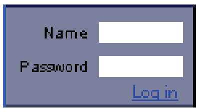
The web server only starts when it has been activated in the properties of the CPU in the "Web server" section.

Note the following:

The web pages are normally transmitted via an non-secure connection and are not secured against hacker attacks. If you want to transfer the web pages in encrypted form to the browser, use the URL https://, followed by the IP address of the CPU.

Logon

No logon is required to access the standard web pages read-only. To execute certain actions, such as changing the operating state of the CPU or for write access, the user must be logged on as "admin". The logon input boxes are on the top left of each standard web page.



If you log on as "admin", you must enter the user name and password there.

Name: admin.

Password: configured CPU password (for password-protected CPU).

JavaScript and cookies

The standard web pages use JavaScript and cookies. You must enable both in your web browser.

If JavaScript is not enabled, the following limitations apply:

- Data from standard web pages are not automatically updated.
- You cannot log on as "admin".
- Fields cannot be sorted (module information)

If cookies are not enabled you cannot log on as "admin".

See also

Access for HTTPS (Page 438)

Settings for operation

Settings for operation

To be able to use the web server of an S7-1200-CPU, you must select the CPU in the network view or the device view and make the following settings in the inspector window under "Properties > General > Web server":

- Enable the web server
- Restricting access to the CPU to HTTPS transmission protocol (encrypted transmission)

  Access via port 80 is then blocked. Communication is only possible via port 443.
- Enabling automatic update of web pages

  The update interval is set by default and cannot be changed. The CPU updates web pages with changing content ((for example, status information or diagnostics information) at regular intervals.

Access for HTTPS

Access via HTTPS

HTTPS is used for encrypting and authentication of communication between the browser and web server.

To transfer data between the browser and the CPU using the HTTPS protocol, enter the URL as https://ww.xx.yy.zz in the address line of your browser, whereby ww.xx.yy.zz stands for the IP address of the CPU.

You require a valid, installed certificate for error-free HTTPS access to the CPU.

If no certificate is installed a warning is displayed with a recommendation not to use this page. To view the page you must explicitly "Add exception".

You can receive a valid certificate (Certification Authority) "SIMATIC CONTROLLER" as a download from the "Intro" web page under "Download certificate". The help function for your respective web browser provides information on how to install a certificate.

Accessing data logs

The "Data logs" web page allows files to be viewed or downloaded that have been created using DataLog instructions and that have been filled with data. You can clear or delete these entries after downloading by logging on as "admin".

Opening a data log

To open a data log, click on the link of the desired data log. You can then open the file (.csv), for example, in Microsoft Excel or in another program you choose or you can save the file.

Special feature: Data logs are saved in U.S. American CSV format. You can only open the file directly using the U.S. version of Microsoft Excel. If you are using another national version of Microsoft Excel, you must import the file, selecting "comma" in the import assistant as the delimiter.

Downloading a data log

To download a data log, click on the download icon of the desired data log. You can then open the file (.csv), for example, in Microsoft Excel or in another program you choose or you can save the file.

Downloading and clearing or deleting a data log

To download and delete the current entries of the data log, you must be logged on as "admin". To do this, click on the "Download and delete" icon of the required data log. You can then open the file (.csv), for example, in Microsoft Excel or in another program you choose or you can save the file. The web server delete the content of the file. The file itself is not deleted, only its content. New data can then be written in this file.

Determining the amount of content

As a default, the 25 most-used entries are displayed, irrespective of how many entries are contained in the data log. The number of displayed entries can be set.

Creating and loading custom web pages

What you need to know about custom web pages

Concept

The concept of custom web pages allows you to access freely-designed web pages of the CPU from a web browser. The web server of the CPU provides this function.

You are not dependent on special tools for the design and functionality of the custom web pages. You can adapt the pages in the layout with CSS, provide dynamic content with JavaScript or use any framework to produce web pages.

The totality of files processed by the web server is also referred to as the "web application".

Web application and user program

Using HTML code in user-defined web pages, you can also transmit data via a web browser to the user program of the CPU for further processing and can display data from the operand area of the CPU in the web browser.

You can use script instructions (such as Javascript) to optimize your web pages, for example to dynamically change contents or validate user entries.

To synchronize between the user program and the web server, but also to initialize, you must call the WWW (SFC 99) instruction in the user program.

- If no interaction is required between the web application and the user program, for example, if a web page only provides static information, only initialization in the user program is required.
- If a simple data exchange is necessary between PLC tags and tags in web applications, to display the contents of PLC tags or write a value in a PLC tag for example, the syntax for reading and writing tags has to be observed. In this case only an initialization is required in the user program, for example in the startup OB.
- If a further interaction is required between the web application and the user program, you must handle status and control information from the Web Control DB in addition to the synchronization between Web server and user program. This is the case, for example, when user entries are transmitted via the web browser to the web server for evaluation by the CPU. Unlike simple data exchange, the user program directly influences the time at which the requested web page is relayed back to the web browser. In this case, you must be acquainted with the concept of manual fragments and the structure of the Web Control DB.

Initialization

Custom web pages are "packaged" in data blocks for processing by the CPU. You must generate appropriate data blocks from the source data (HTML files, images, JavaScript files, etc.) during configuration to be able to download the web application into the CPU. The Web Control DB has a special role (default: DB 333). It contains status and control information as well as links to additional data blocks with coded web pages. Data blocks that contain coded web pages are termed "Fragment DBs".

When the data block is downloaded into the CPU, the CPU does not "know" that custom web pages are coded inside it. The "WWW" (SFC 99) instruction, for example, in the Startup OB informs the CPU which DB is the Web Control DB. The custom web pages can be accessed via a web browser after this initialization.

Synchronization

If the user program is to exchange data with the user-defined web pages, the WWW (SFC 99) instruction must be used in the cyclic program section.

Examples of interaction between user program and web page:

- Check received data
- Assemble and send back data to the web browser making the request

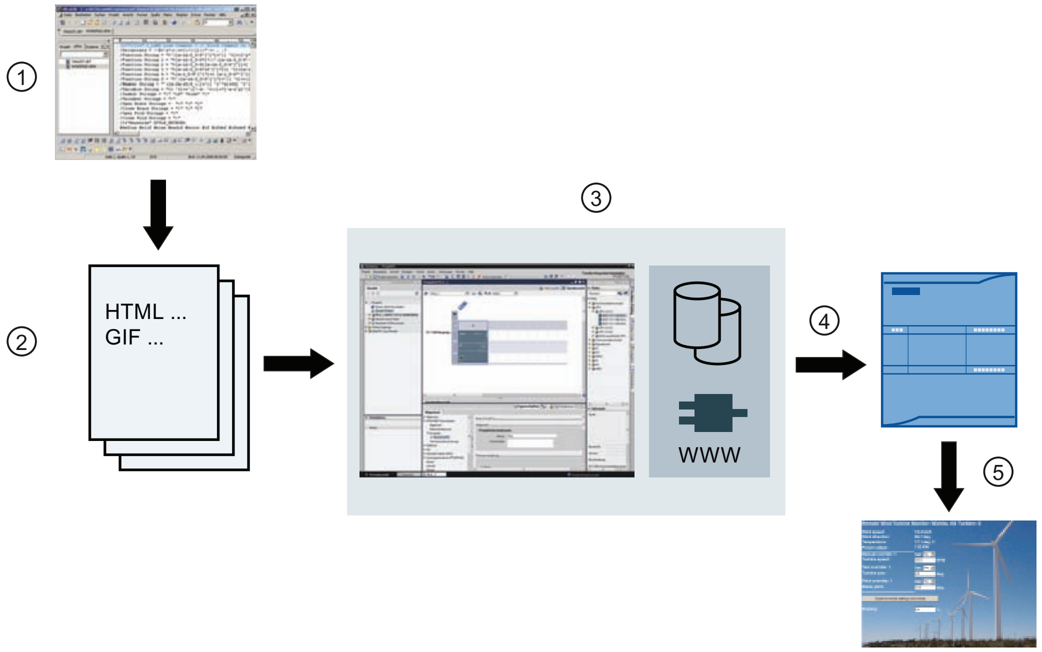
In this case, the status information must be able to be evaluated and control information must be transmitted to the web server, for example, to release a requested web page.

Procedural overview

Basic considerations

This section provides a step-by-step explanation of the basic procedure used to create and download custom web pages and to use them in the operating phase.

The following graphic provides a simplified representation of the process used in creating and displaying custom web pages:



① Programming a web application (using suitable tools when required and AWP commands for dynamic pages when applicable).

② The web application is comprised of single source files, for example, *.html, *.gif, *.js, etc.

③ Using STEP 7:

- Generate the data blocks (Web Control DB and fragment DBs) from source files. The DBs contain meta information and the complete web application, including the images and the dynamic and static parts of the web application. The DBs are stored under "System blocks" in the project tree.
- Call the "WWW" instruction in the user program. This instruction initializes the web server of the CPU for a web application.
- If required, complete final programming for interaction between the web server and user program

④ Downloading the blocks to the CPU.

⑤ Call the web page in the browser. The web pages of the CPU are called by entering the IP address of the CPU.

Additional information

You can find additional information and examples relating to the S7-1200 web server on the Internet (http://support.automation.siemens.com/WW/view/en/36932465).

Creating web pages

Web design tools from various companies can be used to create custom web pages. As a rule, the web pages should be programmed and designed compliant to the conventions of the W3C (World Wide Web Consortium). No check is made for compliance to W3C criteria in the web server of the CPU.

Rules

- The tool must be able to directly edit the HTML code so that the AWP command can be inserted into the HTML page.

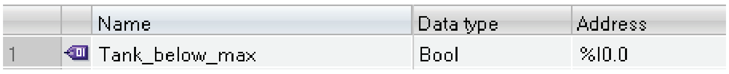
  Only the AWP commands are parsed in the CPU and, for example, replaced by values from the user program/process image of the CPU.
- Files containing AWP commands must be coded in UTF-8. In the metadata of the HTML page, therefore, set the attribute charset to UTF-8 and save the file UTF-8 coded.
- Files containing AWP commands must not contain the following sequence:]]>
- Files containing AWP commands must not contain the following sequence outside of the "Tag-read ranges" (:=<Tag name>:): :=

  Tip: Replace the first character of a prohibited sequence with its character coding; for the colon, for example, &#58;.

A small example for a custom web page should make clear the basic design.

Requirements

- The CPU must have a web server and the web server of the CPU must be activated.
- To be able to access PLC tags with write access as a user, you must be logged on as "admin".
- For the example below, PLC tags must be defined for those PLC tags that are to be shown on the web page. This is shown here for the first tab used, "Tank_below_max".



		Name	Data type	Address
1		Tank_below_max	Bool	%I0.0

Creating custom web pages

The following code for an example web page reads values from the process image and provides them in a table.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-
8">
   <title>Mix</title>
  </head>
   <body>
    <h1>Mix</h1>
    <h2> Actual State </h2>
    <table border="1">
     <tr>
      <th>Variable</th>
      <th>State</th>
     </tr>
     <tr>
      <td>Tank below max</td>
      <td>:="Tank_below_max":</td>
     </tr>
     <tr>
      <td>Tank above min</td>
      <td>:="Tank_above_min":</td>
     </tr>
    </table>
   </body>
</html>
```

AWP commands

The interface between a freely-programmable web application for a CPU that has a web server and the CPU data is declared by the AWP command (Automation Web Programming).

To develop web applications you are only subject to the restrictions of the web browser. In one of the programming languages of STEP 7, control with the user program which CPU data is displayed at what time in the web browser of the viewer. Use AWP commands, which you comment within the HTML files, to declare data to be used for intentional interaction between the web application and the user program.

AWP commands are inserted as HTML comments with a special syntax into HTML files; they declare the following features:

- Read PLC tags
- Write PLC tag
- Read special tags
- Write special tags
- Define enum types
- Assign tags to enum types
- Defining fragments
- Import fragments

Syntax of AWP commands

An AWP command begins with "`<!--AWP_`" and ends with "`-->`". In JavaScript files, the commands should also be enclosed by JavaScript comments ("`/*...*/`").

Notation rules for PLC tag names within an AWP command

The AWP commands "AWP_In_Variable" and "AWP_Out_Variable" contain a name attribute and optionally a use attribute. A PLC tag name is assigned to these attributes, by means of which the PLC tags in the browser are written or read. The following rules apply to handling PLC tag names in HTML code:

- PLC tags must be enclosed in quotation marks (" ... ").
- PLC tags used in AWP commands must also be enclosed by single quotation marks ("' ... '") or with quotation marks masked by a backslash ("\" ... \"").
- If the PLC tag name contains the character \ (backslash), this character must be designated with the escape sequence \\ as standard character of the PLC tag name.
- If the PLC tag name in the AWP command is also enclosed by single quotation marks and the single quotation mark (') occurs within the name, it must also be designed as normal character by the escape sequence \'.
- If an absolute address (input, output, bit memory) is used in AWP command, it is enclosed by single quotation marks.

PLC tag	**PLC tag in HTML code**
`"Velocity"`	`<!-- AWP_In_Variable Name='"Velocity"' -->`
	`<!-- AWP_In_Variable Name="\"Velocity\"" -->`
`"abc\de"`	`<!-- AWP_In_Variable Name='"abc\\de"' -->`
`"abc'de"`	`<!-- AWP_In_Variable Name='"abc\'de"' -->`
`"abc'de"`	`<!-- AWP_In_Variable Name="abcde" Use'"abc\'de"' -->`
`"DB name".tag`	`<!-- AWP_In_Variable Name='"DB name".tag' -->`
`"DB name"."ta.g"`	`<!-- AWP_In_Variable Name='"DB name"."ta.g"' -->`
`-`	`<!-- AWP_Out_Variable Name=’flag1’ Use='M0.0' -->`

See also

Reading tags (Page 445)

Writing tags (Page 447)

Special tags (Page 448)

Reading tags

Custom web pages can read PLC tags.

The PLC tag must be specified by a PLC tag name.

These OUT variables (direction of output as viewed from the controller) are inserted at any location within the HTML text with the syntax described in the following.

Syntax

`:=<varname>:`

These references are replaced when the web server is in operation by the current values of the PLC tag in each case.

`<varname>` can be a simple, global CPU tag but also a complete tag path to a structure element.

Notation rules for PLC tag names

- PLC tags in HTML code are enclosed by quotation marks ("), if they are defined in the tag table. In the case of data block tags, the name of the data block is enclosed by quotation marks. If special characters are used in the structure elements of the data block, for example the dot (.) or blank, this part must also be enclosed by quotation marks.
- Quotation marks are not used for absolute addresses of inputs, outputs or bit memories.

PLC tag	PLC tag in HTML code
"DB_name".var_name	:="DB_name".var_name:
"DB_name".struct_name.var_name	:="DB_name".struct_name.var_name:
"DB_name"."var.name"	:="DB_name"."var.name":
"memory"	:="memory":
-	:=I0.0:
	:=Q0.0:
	:=MW100:
	:=%MW100:
"My_Data_Block".flag1	<!-- AWP _Out_Variable Name=’flag1’ Use='"My_Data_Block".flag1' -->
	...
	:=flag1:

- If the PLC tag name contains the character : (colon) or \ (backslash), this character must be designated with the escape sequence \: or \\ as standard character of the PLC tag name.

PLC tag	PLC tag in HTML code
"abc:de"	:="abc\:de":
"abc\de"	:="abc\\de":

- Special characters "<, &, >"

  Display problems can occur if these characters are contained in the tag name (for example, "a<b").

  Avoid expressions such as :="a<b": in the HTML page.

  To prevent display problems from occurring, use e.g. an AWP command with a use expression according to the pattern depicted below. The use attribute defines the PLC tag with the problematic character, the name attribute defines the name without problematic character, as it is used in the HTML page.

PLC tag	PLC tag in HTML code
"a<b"	<!—AWP _Out_Variable Name=’simplename’ Use='"a<b"' -->
	...
	:=simplename:

See also

AWP commands (Page 444)

Writing tags

Custom web pages can write data into the CPU.

This requires an AWP command that identifies the PLC tag to be written.

The PLC tag must also be specified by a PLC tag name.

The IN tags (direction of input as viewed from the controller) are placed on the browser page. This can be done, for example, in a form.

The tags are either set in the HTTP header (by cookie or POST method) or in the URL (GET method) by the browser and are then written by the web server into the respective PLC tag.

Syntax

To allow the IN tags to be written to the CPU, the tags must first be defined by an explicit AWP instruction:

```
<!-- AWP_In_Variable Name='<PLC_Varname1>' Name='<PLC_Varname2>'
Name='<PLC_Varname3>' -->
```

Several tags can be defined in an instruction - such as that shown above.

The specific PLC tag name is hereby written in double quotation marks; for example <PLC_Varname1> = "myVar".

In cases where the name of the tag that you use for the web application is not identical to the name of the PLC tag, the "Use" parameter can be used to assign to a PLC tag.

```
<!-- AWP_In_Variable Name=’<Webapp_Varname>’ Use=’<PLC_Varname>’
```

Example

The "AWP_In_Variable" AWP command is indispensable when handling forms.

```
<form method='post' action='/awp/appl/x.html'>
 <p>
  <input name='"var1"' type='text'>
  <input value='set' name='Button1' type='submit'>
 </p>
</form>
```

In the form defined above, the HTTP request method "post" is used to transfer the tag "var1" to the web server. The user places the "var1" tag in the form field. The tag 'Button1' has the value 'set', but is not required for the CPU. To allow the "var1" tag to be written to the CPU, the following instruction must be included in the same fragment:

```
<!-- AWP_In_Variable Name='"var1"' -->
```

Since PLC tags are enclosed in double quotation marks ("), the name in the AWP command must be enclosed in single quotation marks (') or in masked quotation marks (\"). To avoid the numerous escape sequences, we recommend the use of single quotation marks.

```
<!-- AWP_In_Variable Name=’"Info".par1’ -->
<!-- AWP_In_Variable Name="\"Info".par1\"" -->
```

Conditions for write access during operation

The following requirements have to be met in order for a user to be able to write to PLC tags from a user-defined web page.

- The CPU is is password protected.
- The user is logged in as "admin".

This rules applies to all writing access to web pages on a CPU.

See also

Requirements for web access (Page 436)

AWP commands (Page 444)

Special tags

Special tags are mainly HTTP tags set in the definition of the World Wide Web Consortium (W3C) . Special tags are also used for cookies and server tags.

The AWP command to read and write special tags differ only in that they have additional parameters than the AWP command used to read and write normal tags.

Reading a special tag

The Web server can read PLC tags and transfer these to special tags in the HTTP Response Header. You can, for example, read a URL for a diversion to another web page and transfer to the special tag HEADER:Location using the special tag HEADER:Location.

The following special tags can be read:

Name	Description
COOKIE_VALUE:name	Value of cookie with name: "name"
COOKIE_EXPIRES:name	Execution time of cookie with name: "name" in seconds (must be set beforehand).
HEADER:Status	HTTP status code (if no other value has been set, status code 302 is returned).
HEADER:Location	Path for forwarding to another page. Status code 302 must be set.
HEADER:Retry-After	Anticipated time in which the service is not available. Status code 503 must be set.
HEADER: …	All other header tags can also be forwarded in this way.

Use the AWP command "AWP_Out_Variable" to specify which PLC tags are to be transferred in the HTTP header to the web browser.

Basic structure:

```
<!-- AWP_Out_Variable Name="<Typ>:<Name>" [Use="<Varname>"] -->
```

Parameter description

- Name: Type and name of special tag
- Use (optional parameter): In cases where the name of the special tag is not identical to the name of the PLC tag, parameter "Use" can be used to assign to a PLC tag.

Example:

```
<!-- AWP_Out_Variable Name="COOKIE_VALUE:siemens" Use='"info".language' --
>
```

Writing a special tag

In principle, all HTTP tags written in the HTTP header by the web browser can be evaluated by the user program of the CPU. Examples of tag types:

Name	Description
HEADER:Accept-Language	Accepted or preferred language
HEADER:Authorization	Proof of authorization for a requested resource
HEADER:Host	Host and port of the requested resource
HEADER:User-Agent	Information on the browser
HEADER: …	All other header tags can also be forwarded in this way
SERVER:current_user_id	Indicates whether a user is logged on (current_user_id=0: no user logged on)
SERVER:current_user_name	User name of the user logged on
SERVER:GET	Request method is GET
SERVER:POST	Request method is POST
COOKIE_VALUE:name	Value of cookie with name: "name"

The AWP command "AWP_In_Variable" is used to define which special tags are to be evaluated in the user program of the CPU.

Basic structure:

```
<!-- AWP_In_Variable Name="<Typ>:<Name>" [Use="<Varname>"] -->
```

Parameter description:

Name: Type and name of special tag

Use (optional parameter): In cases where the name of the special tag is not identical to the name of the PLC tag, the parameter Use can be used to assign to a PLC tag.

Examples:

```
<!-- AWP_In_Variable Name="COOKIE_VALUE:siemens" Use='"info".language' -->
```

The tag name in the HTTP header is replaced by the PLC tag name specified by Use . The cookie is written to the PLC tag "info".language .

```
<!-- AWP_In_Variable Name='COOKIE_VALUE:siemens' Use='"info".language' -->
```

The tag name in the HTTP header is replaced by the PLC tag name specified by Use. The cookie is written to the PLC tag "info".language .

```
<!-- AWP_In_Variable Name='"COOKIE_VALUE:siemens"' -->
```

The HTTP-header variable is written in the same-name PLC variable.

See also

AWP commands (Page 444)

Enumeration types

Enumeration types (enums)

Numerical values from the PLC program can be converted into text and vice versa using enums. The numerical values can also be assigned for several languages.

Creating enums

Enter an AWP command using the following syntax at the start of the HTML file:

```
<!-- AWP_Enum_Def Name="<Name of the enum type>"
Values='0:"<Text_1>", 1:"<Text_2>", ... , x:"<Text_x>"' -->
```

For example, for German values to be saved as an HTML file in the "de" folder of the HTML directory:

```
<!-- AWP_Enum_Def Name="Enum1" Values='0:"an", 1:"aus", 2:"Störung"' -->
```

For example, for English values, to be saved as an HTML file in the "en" folder of the HTML directory:

```
<!-- AWP_Enum_Def Name="Enum1" Values='0:"on", 1:"off", 2:"error"' -->
```

Assigning enums

Tags are assigned from the user program to the individual enum texts using a special AWP command:

```
<!-- AWP_Enum_Ref Name="<VarName>" Enum="<EnumTypeName>" -->
```

<VarName> is thereby the symbolic name from the user program and <EnumTypeName> is the previously set name of the enum type.

Note

In each fragment in which enum texts are referenced by a PLC tag, this PLC tag must be assigned by the appropriate AWP command of the enum type name.

Example

Enum type "state" is defined with values "0" and "1". "0" means "off", "1" means "on":

```
<!-- AWP_Enum_Def Name="state" Values='0:"off", 1:"on"' -->
```

The following code is contained in the HTML code of the web page to be output:

```
<!-- AWP_Enum_Ref Name="operating state" Enum="state" -->
:=operating state:
```

Depending on the value of the "operating state" tag, the ´result displayed is no longer "0" or "1", but "off" or "on".

Definition of fragments

Fragments

Fragments are "logical sections" of a web page to be processed by the CPU individually.

Fragments are usually complete pages but can also be individual elements such as files (for example, images) or complete documents.

Defining fragments

```
<!-- AWP_Start_Fragment Name="<Name>" [Type="<Type>"] [ID="<Id>"] -->
```

The start of a fragment is specified by this command. A fragment runs to the start of the next fragment or to the end of the file.

- <Name> Indicates the name of the fragment.

  The name must start with a letter [a-zA-Z] or an underscore (_). Letters, underscores or numbers [0-9] can follow after this first character.
- <Type> Indicates the type of the fragment.
  - "manual" The user program is informed of the request for a fragment; the web page to be returned can be changed by the user program.
  - "automatic" The page is automatically processed (default).
- <id> A numeric ID can be stipulated for the fragment. If no ID is assigned, the fragment is automatically assigned an ID. For manual pages (<Type>=manual) , the fragment can be addressed in the user program of the CPU by this ID.

Note

Keep the ID low because the highest ID influences the size of the Web Control DB.

The input document is completely divided into fragments by the "AWP_Start_Fragment" command. "AWP_End_Fragment" is therefore unnecessary.

Without a start fragment command, a file is mapped as a fragment; the fragment name is derived from the file name. If a file is divided into several fragments (by "AWP_Start_Fragment"), the file must begin with the "AWP_Start_Fragment" command.

Importing fragments

You can declare a fragment in an HTML page and import this fragment into other web pages.

Example

A company logo is to be displayed on all web pages of a web application.

There is only one instance of the HTML code for the fragment that displays the company logo. You can import the fragment as often and into as many HTML files as required.

Syntax

```
<!-- AWP_Import_Fragment Name = "<name>"-->
```

- <name> is the name of the fragment to be imported.

Example

HTML code within a web page that declares a fragment:

```
<!-- AWP_Start_Fragment Name = "My_Company_Logo"-->
<p><img src = "compay_logo.jpg"></p>
```

Example

HTML code within another web page that imports the declared fragment:

```
<!-- AWP_Import_Fragment Name = "My_Company_Logo"-->
```

Creating and loading a data block

Requirement

- All source files required for the web application have been created.

Procedure

To create data blocks from the source files for custom web pages in STEP 7, proceed as follows:

1. Select the CPU, for example, in the device configuration.
2. Select the properties for custom web pages in the inspector window under "Properties > General > Web server".
3. As "HTML source", select the folder that contains the source files for the web application.
4. Enter the HTMP page to be opened on starting the web application as the start HTML page.
5. Enter a name for the application if required.
6. You can supplement a range of file name extensions as "Files with dynamic content" if necessary. Only enter those file name extensions that also contain AWP commands.
7. The number for the Web Control DB and for the fragment DB start number can be kept as long as they are not already being used by your user program.

8. Click on the "Generate" button to create DBs from the source files.

   The generated data blocks are saved in the project navigation in the "System block" folder (in the "Web server" subfolder).

9. In the CPU, select the network view to be loaded and then select the "Download to device" command in the "Online" menu to download the blocks. Compilation of the blocks is implicitly initiated before downloading.

   If errors are reported during this process, you must correct these errors before you can download the configuration.

Structure of the PLC program

Your user program must call the " WWW" instruction to even allow the web application, for example, the custom web pages, to be available to the CPU on the standard web pages and to allow them to be called up there.

The Web Control DB you have created from the source files is the input parameter (CTRL_DB) for the "WWW" instruction. The Web Control DB references the content of the custom web pages coded in the fragment DB and then receives status and control information.

Calling the "WWW" instruction in the startup program

If you do not want the user program to influence requested web pages, it is sufficient to only call the "WWW" instruction once in a startup OB. This instruction initializes communication between the web server and the CPU.

Calling the "WWW" instruction in the cyclic program

The "WWW" instruction can also be called in an OB processed in cycles (for example, OB 1). This has the advantage of being able to respond to web server requests from within the user program. Manual fragments must be used for this.

In this case, you must evaluate information from the Web Control DB in order to identify the requested web page or the requested fragment. On the other hand, you must set a bit in the user program in order to explicitly release the web page to be returned by the web server after processing the web page request.

The structure of the Web Control DB is described in the following section.

Web Control DB

The Web Control DB (DB 333 by default) is created by STEP 7 and contains information on the structure of user pages, the status of communication and any errors that occur.

Additional fragment DBs are also created as well as the Web Control DB. These fragment DBs (there may also only be one fragment DB) are referenced in the Web Control DB. The fragment DBs contain the web pages and media data coded in fragments, for example, images. The content of the fragment DB cannot be changed by the user program. It is created automatically and is only for data management.

The status and control tags of the Web Control DB are accessed via symbols.

The following lists the tags of the Web Control DB required for status evaluation and to control interaction.

The Web Control DB provides two types of information:

- Global status information: Not bound to a concrete web page request.
- Request status and control information: Information about queued requests.

Global status information

"WEB-Control_DB".commandstate.init	Activates and initializes the web application.
"WEB-Control_DB".commandstate.deactivate	Deactivates the web application.
"WEB-Control_DB".commandstate.inititializing	The web application is initialized (read Web Control DB, etc.).
"WEB-Control_DB".commandstate.error	Web application could not be initialized. The reason is coded in "WEB-Control_DB".commandstate.last_error .
"WEB-Control_DB".commandstate.deactivating	The web application is closed.
"WEB-Control_DB".commandstate.initialized	The web application has been initialized and is ready.
"WEB-Control_DB".commandstate.last_error	Refer to the next table for a value table of possible errors.

Last_error	Description
1	Fragment DB is inconsistent (does not match the Web Control DB).
2	A web application already exists with this name.
3	Memory problem initializing in the web server.
4	Inconsistent data in the Web Control DB.
5	A fragment DB is not available (not loaded).
6	No AWP ID for a fragment DB.
7	The enum fragment is not available (contains the texts and information on the enum types).
8	An action requested via the command flag in the Web Control DB is prohibited in the current state.

Last_error	Description
9	Web application is not initialized (if there is no reinitializing after disabling).
10	Web server is disabled.
...	Last_error is reset once the web application has been successfully initialized.

Request status information

Request status information is bound to one of four possible requests, x = [1 … 4].

"WEB-Control_DB".requesttab[x].idle	Nothing need be done.
"WEB-Control_DB".requesttab[x].waiting	The user program must react to a request from a manual fragment and explicitly initiate further processing in the web browser.
"WEB-Control_DB".requesttab[x].sending	The web server is occupied with processing the request/fragment.
"WEB-Control_DB".requesttab[x].aborting	The TCP connection is closed by the web server.

Request control information

Request control information is bound to one of four possible requests, x = [1 … 4].

"WEB-Control_DB".requesttab[x].continue	Releases the fragment being processed for transmission. Processing of the next fragment is initiated.
"WEB-Control_DB".requesttab[x].repeat	Releases the fragment being processed for transmission. The fragment is then processed again.
"WEB-Control_DB".requesttab[x].abort	Closes the TCP connection.
"WEB-Control_DB".requesttab[x].finish	Releases the fragment being processed for transmission. Stops further processing of requests (terminates the request).

Example:

The tag for the DB is: "WEB-Control_DB". Whether errors have occurred during initialization of the web application can be determined by requesting bit "WEB-Control_DB".commandstate.error in the user program.

If an error has occurred you can analyze it using the "WEB-Control_DB".commandstate.last_error value.

Interaction with the user program

Using manual fragments ensures that the user program reacts synchronously to the user program, thereby allowing the responding web page to be processed by the user program.

Fragment type

To react to the received data in the user program the "manual" fragment type must be used for the fragment writing the data (for "manual pages"):

```
<!-- AWP_Start_Fragment Name="testfrag" ID="1" Type="manual" -->
```

The values are always transferred to the web server of the CPU for automatic and manual pages in the same way:

Example:

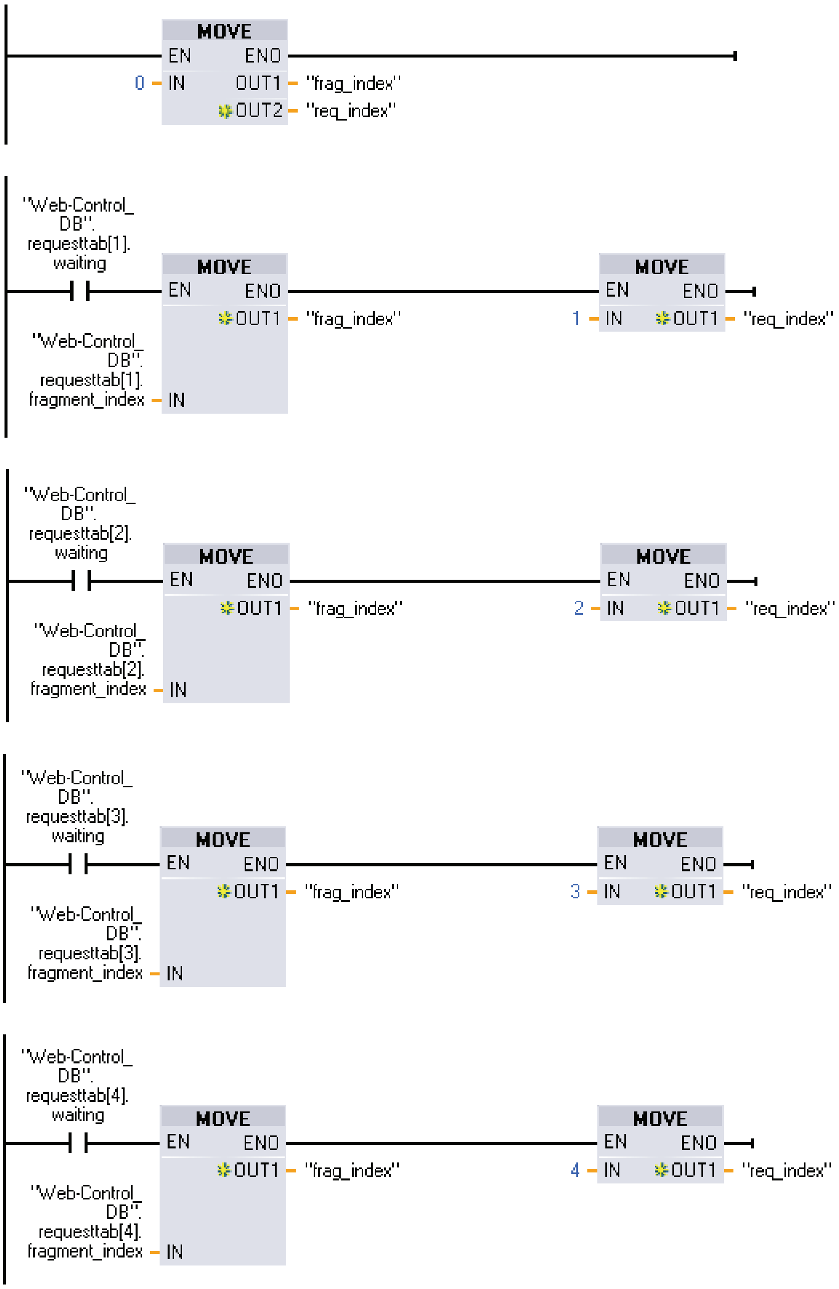
```
<form method="POST" action="">
<p>
<input type="submit" value="Set new value">
<input type="text" name='"Velocity"' size="20">
</p>
</form>
```

User program for manual fragments

When using manual pages, the "WWW" instruction must be called in cycles in the user program of the CPU.

To react to values entered in the browser, the request - which is made by the manual page to the web server - must first be evaluated in the user program. To do this, the Web Control DB (for example, DB 333) must be examined in cycles for queued requests. The array that manages four requests is contained in the "requesttab" section of the Web Control DB. Each element of the array thereby contains information on the respective request within a structure.

A simple programming example shows how queued requests are checked based on the tags of the Web Control DB.

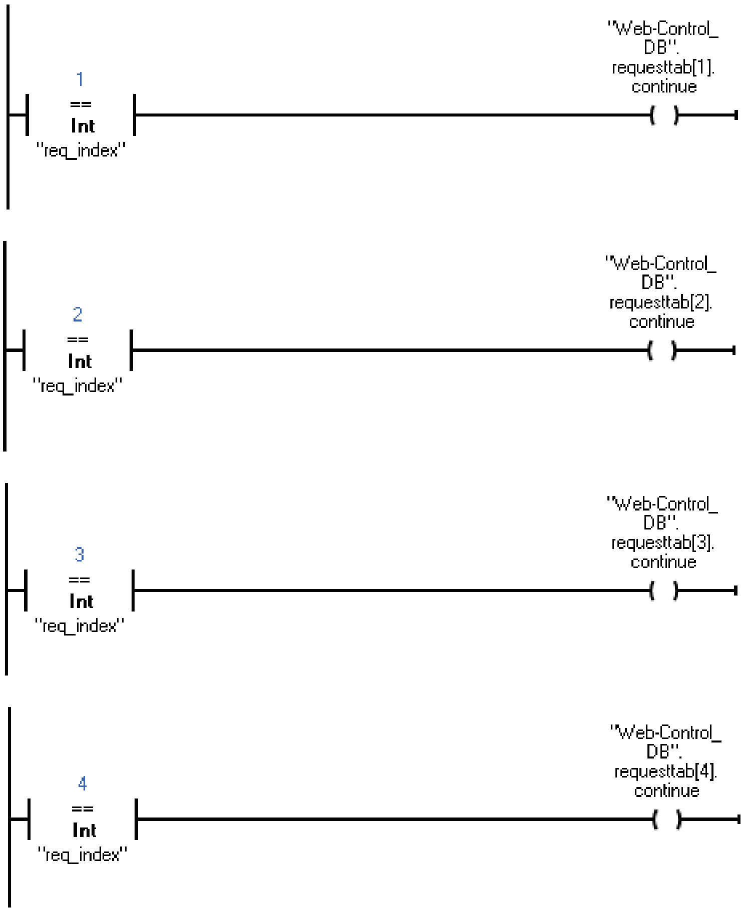

MOVE
EN
ENO
0
IN
OUT1
"frag_index"
OUT2
"req_index"
"Web-Control_DB".requesttab[1].waiting
MOVE
EN
ENO
OUT1
"frag_index"
"Web-Control_DB".requesttab[1].fragment_index
IN
MOVE
EN
ENO
1
IN
OUT1
"req_index"
"Web-Control_DB".requesttab[2].waiting
MOVE
EN
ENO
OUT1
"frag_index"
"Web-Control_DB".requesttab[2].fragment_index
IN
MOVE
EN
ENO
2
IN
OUT1
"req_index"
"Web-Control_DB".requesttab[3].waiting
MOVE
EN
ENO
OUT1
"frag_index"
"Web-Control_DB".requesttab[3].fragment_index
IN
MOVE
EN
ENO
3
IN
OUT1
"req_index"
"Web-Control_DB".requesttab[4].waiting
MOVE
EN
ENO
OUT1
"frag_index"
"Web-Control_DB".requesttab[4].fragment_index
IN
MOVE
EN
ENO
4
IN
OUT1
"req_index"

In cases where a request has been made, this program section writes the fragment ID in the #frag_index tag and the request no. (value range 1-4) in the #req_index tag.

Using the information from this, the information transferred in the request can now be processed separately for each fragment ID in the program (for example, plausibility check).

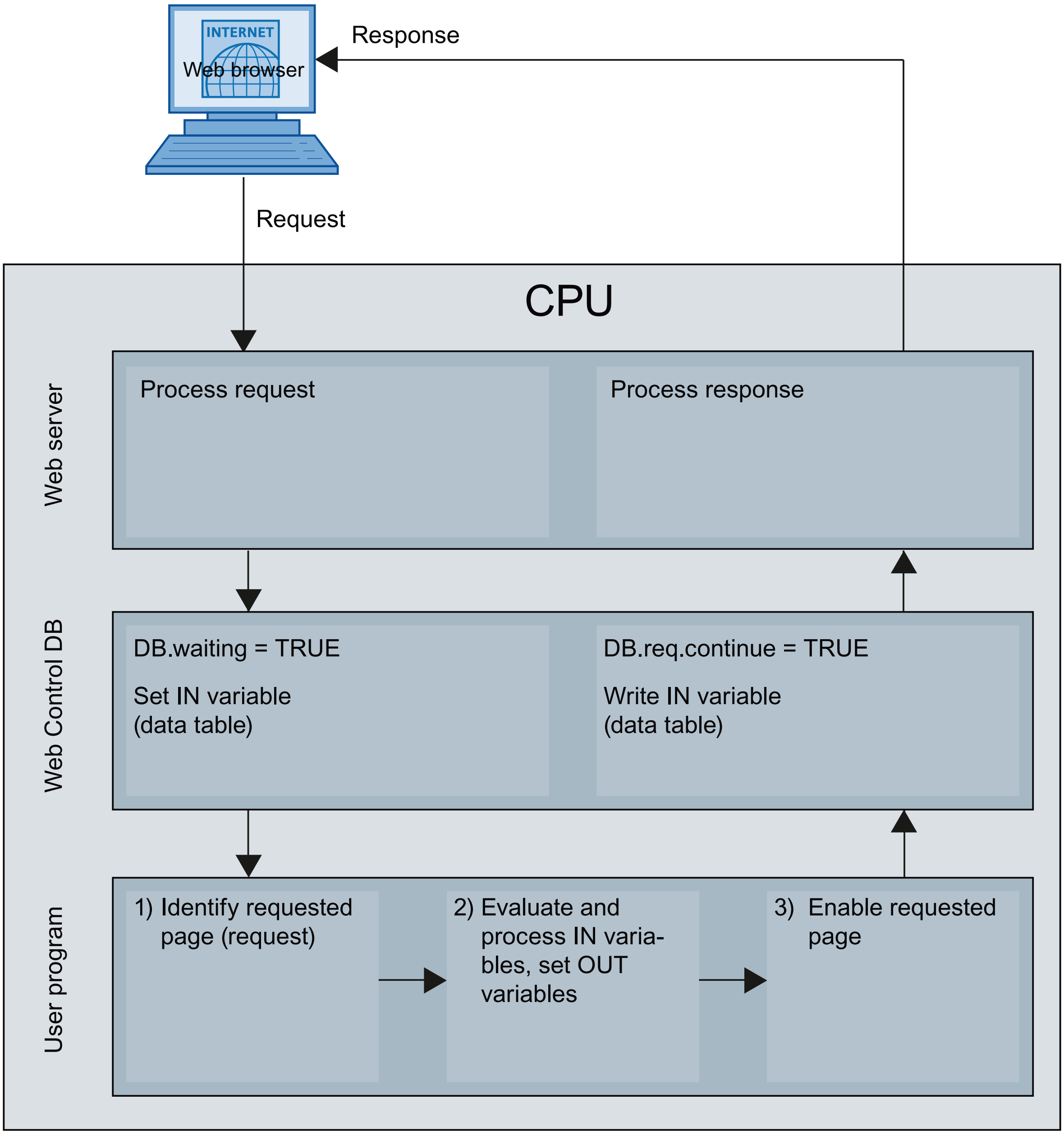
Once processing of the request has been completed by the program, the request must be answered and the appropriate entry is once more reset under"requesttab" of the Web Control DB (for example, DB 333).

A simple programming example for replying to requests:


"Web-Control_
DB".
requesttab[1].
continue
1
==
Int
"req_index"
"Web-Control_
DB".
requesttab[2].
continue
2
==
Int
"req_index"
"Web-Control_
DB".
requesttab[3].
continue
3
==
Int
"req_index"
"Web-Control_
DB".
requesttab[4].
continue
4
==
Int
"req_index"

Principle sequence of a browser request with interaction from the user program

The following figure shows the simplified, principle sequence of the web browser request on the effects of Web Control DB content and the actions required from the user program until the processed web page is returned (response).



Displaying custom web pages in the browser

Display web pages in browser

Web pages are called from the standard web pages of the web browser.

In addition to the other links in the navigation bar, the standard web pages also have a link to "user pages".

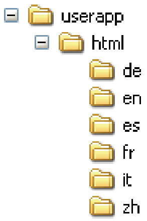
Click on the "user pages" link to open the web browser you have configured as the default HTML page.

Creating custom web pages in several languages

You can make each of your custom web pages available in various languages.

Requirements

The language-dependent HTML; pages must be stored in a folder structure containing folders with the respective language abbreviations:



Specified language abbreviations

Language abbreviations "de", "en", "fr", "es", "it" and "zh" are fixed. Additional language folders or other designated language folders are not supported.

Additional folders within the same folder hierarchy for other files can be created as required; for example, an "img" folder for images and a "script" folder for JavaScript files.

Language switching for custom web pages

Requirements

The HTML pages are contained in the predefined language folders, for example, HTML pages with German text are in the "de" folder, HTML pages with English text are in the "en" folder.

Language switching concept

Language switching is based on a predefined cookie named "siemens_automation_language". If the cookie is set to value "de", at the next web page request or web page update, the web server switches to the web page from the "de" folder.

Similarly, the web server switches to the web page from the "en" folder when the cookie is set to "en".

Example of language switching

The example is structured as follows:

- The language-dependent HTML files with the same name, for example, "langswitch.html" are located in both language folders "de" and "en". The text to be displayed within the two files are German or English, corresponding to the name of the folder.
- There is an additional "script" folder in the folder structure containing the JavaScript file "lang.js". Functions required for language switching are stored in this file .

Structure of the "langswitch.html" file ("de" folder)

Meta data "content language", charset and path to JavaScript file are set in the file header.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Language" content="de">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Switch language to German page</title>
<script type="text/javascript" src="script/lang.js" ></script>
<head>
```

Language selection is implemented in the body of the file by the "select" HTML element. The select element initiates a list box and contains the "de" option, labeled as "German" and "en", labeled as "English"; "de" is the default.

The "DoLocalLanguageChange(this)" function is called using the "onchange" event handler. The "this" parameter transmits the select object with the selected option to this function. "onchange" calls the function each time the option is changed.

```
<!-- Language Selection -->
<table>
    <tr>
        <td align="right" valign="top" nowrap>
          <!-- change language immediately on change of the
selection -->
          <select name="Language"
onchange="DoLocalLanguageChange(this)" size="1">
                <option value="de" selected >Deutsch</option>
                <option value="en" >English</option>
            </select>
        </td>
     </tr>
</table>
<!-- Language Selection End-->
```

Structure of the "langswitch.html" file ("en" folder)

The header of the HTML file with English text is structured similarly to the HTML file with German text.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Language" content="en">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Language switching english page</title>
<script type="text/javascript" src="script/lang.js" ></script>
```

Language selection is also implemented in the body of the file by the "select" HTML element. In contrast to the German HTML file, the English option is already selected as a default and the test or the labels are in English.

```
<!-- Language Selection -->
<table>
    <tr>
        <td align="right" valign="top" nowrap>
          <!-- change language immediately on change of the
selection -->
          <select name="Language"
onchange="DoLocalLanguageChange(this)" size="1">
                <option value="de" >German</option>
                <option value="en" selected >English</option>
            </select>
        </td>
     </tr>
</table>
<!-- Language Selection End-->
```

Structure of "lang.js" file (in the "script" folder)

The "DoLocalLanguageChange" function is defined in the JavaScript file; this in turn calls the "SetLangCookie" function with the value of the language selection. SetLangCookie combines the cookie name and the cookie value and then sets the cookie using the appropriate document.cookie property. The web page must then be reloaded (top.window.location.reload) to allow the web server to react to the setting of the cookie by displaying the required language.

```
function DoLocalLanguageChange(oSelect) {
        SetLangCookie(oSelect.value);
        top.window.location.reload();
    }

 function SetLangCookie(value) {
        var strval = "siemens_automation_language=";
            // this is the cookie by which the web server
            // detects the desired language
            // this name is required by the web server
        strval = strval + value;
        strval = strval + "; path=/ ;";
            // set path to the application, since otherwise
            // path would be set to the requesting page
```

```
              // would not get the cookie.
              // The path for user defined applications follows this
sample:
              // path=/awp/<application name>/<pagename>
              // example: path=/awp/myapp/myappstartpage.htm
              //(where myapp is the name of the web application
              // entered in the web server properties of the cpu)
              /*
              use expiration if this cookie should live longer
              than the current browser session
              var now      = new Date();
              var endttime = new Date(now.getTime() + expiration);
              strval = strval + "; expires=" + endttime.toGMTString()
+ ";";
              */
          document.cookie = strval;
      }
```

Additional configurations

Configuring additional functions

The S7-1200 automation system has numerous additional functions that are useful as integrated CPU functions or available via plug-in modules (for example, communication modules). You can find the description via the following links.

See also

Overview of point-to-point communication (Page 560)

General information on high-speed counters (Page 556)

8.1.4.2 Configurations for PROFIBUS DP

The basics of configuring a DP master system

Distributed I/O

DP master systems that consist of a DP master and DP slaves which are connected via a bus and communicate with one another via the PROFIBUS DP protocol are referred to as distributed I/O.

Firmware version of the S7-1200 CPU

Use of the PROFIBUS functions with the S7-1200, requires CPUs with firmware version 2.0 or higher.

Configuring distributed I/O

Since DP masters and DP slaves are different devices, these instructions only provide a basic configuration procedure. However, the process for configuring distributed I/O is practically identical to that of non-distributed configuration.

Creating the DP master system in the network view

After you have used dragged a DP master and a DP slave (for example, a CM 1243-5 and a CM 1243-5) from the hardware catalog to the network view, connect them both to a PROFIBUS subnet.

Additional information

Observe additional information on the scope of functions in the manuals of the respective device.

DP slaves within the hardware catalog

DP slaves within the hardware catalog

You will find the DP slaves in the "Distributed I/O" folder of the hardware catalog. Compact and modular DP slaves are located there:

- Compact DP slaves
  Modules with integrated digital/analog inputs and outputs, for example, ET 200L
- Modular DP slaves
  (Interface modules with S7 modules assigned, for example, ET 200M

The available DP master and the desired functionality will determine which DP slaves can be used.

I slaves within the hardware catalog

The CM 1242-5 is, for example, an DP slave that can be configured as intelligent DP slave. You can find it in the hardware catalog at:

- CM 1242-5
  "PLC > SIMATIC S7 1200 > Communication module > PROFIBUS"

DP/DP coupler in the hardware catalog

Introduction

A DP/DP coupler connects two PROFIBUS DP networks as a gateway so that the DP master from one network can transfer data to the DP master of the other network.

The maximum amount of data that can be transferred is 244 bytes input data and 244 bytes output data.

DP/DP coupler in the hardware catalog

Details of a DP/DP coupler as gateway between two DM master systems are contained in the hardware catalog in the folder "Other field devices > PROFIBUS-DP > Gateways".

Configuring the DP/DP coupler

DP/DP couplers are configured in both PROFIBUS networks, each in their own master systems.

The input and output areas of both networks must thereby be matched to one another. The output data from one end of the DP/DP coupler are accepted as input data at the other respective end and vice versa.

Configurations involving PROFIBUS DP

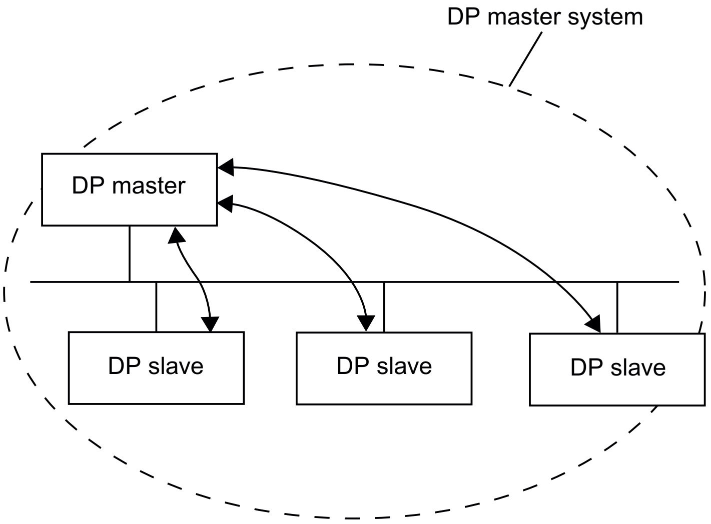
Configurations involving basic DP slaves

Communication between DP master and DP slave

In the case of a configuration involving simple DP slaves, data is exchanged between the DP master and simple DP slaves, i.e. with I/O modules via the DP master. The DP master polls each of the configured DP slaves in its polling list within the DP master system in succession, sending the output data to the slaves and receiving the input data back from the slaves.

Mono-master system

The configuration with only one DP master is also described as mono-master system. A single DP master with its associated DP slaves is connected to a physical PROFIBUS DP subnet.



Configurations involving intelligent DP slaves

Definition

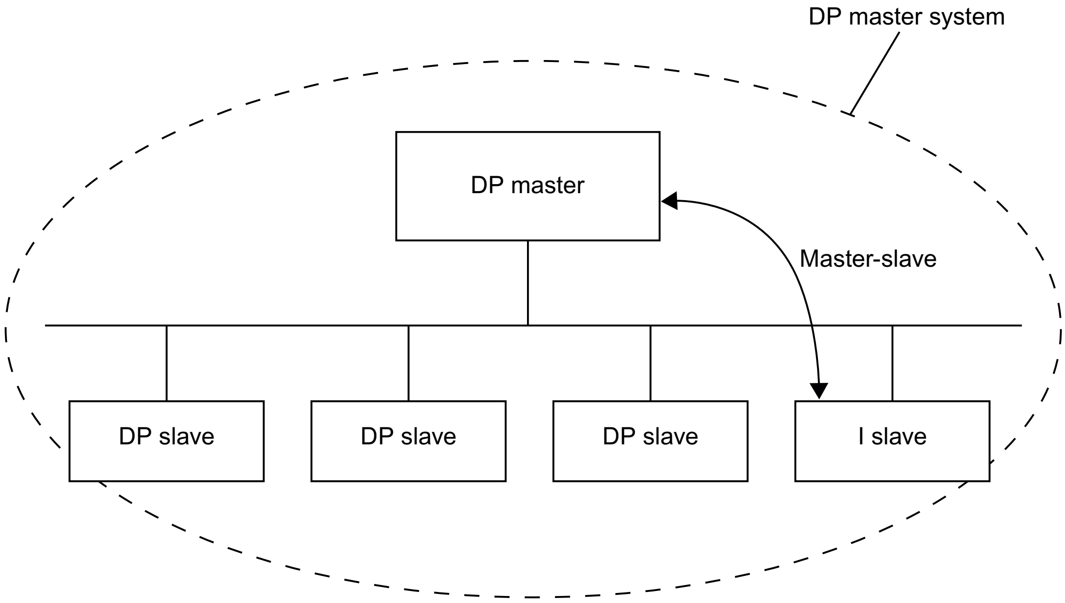
DP slaves that feature their own preprocessing program are referred to as intelligent DP slaves (I slaves).

CM 1242-5 is an intelligent DP slave.

I slave <> DP master data exchange

A higher-level automation system processes the automation task, which is broken down into sub-tasks. The necessary control tasks are processed separately and efficiently in a CPU as preprocessing programs.

In the case of configurations involving intelligent DP slaves, the DP master only accesses the address area of the I slave CPU, and not the I/O modules of the intelligent DP slave. The address area must not be assigned to any actual I/O modules on the I slave. The assignment must be made during I slave configuration.



Configuring distributed I/O systems

Hint: Quick configuration of master systems

If the DP master system has several DP slaves, use drag-and-drop operation to assign to the master in one step all DP slaves that were placed.

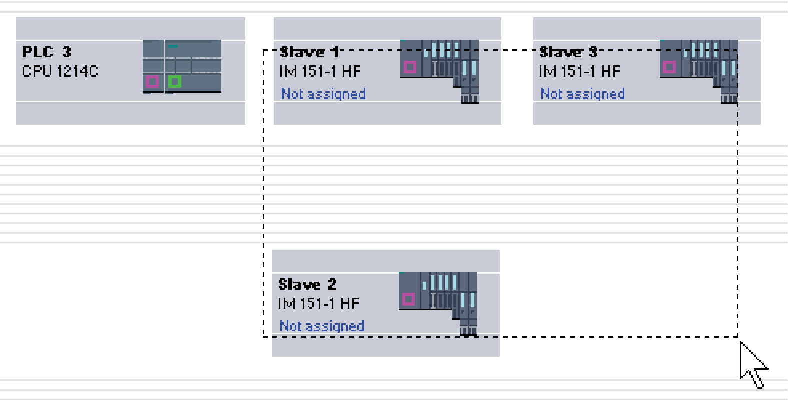
Requirements

DP master and DP slaves are placed in the network view.

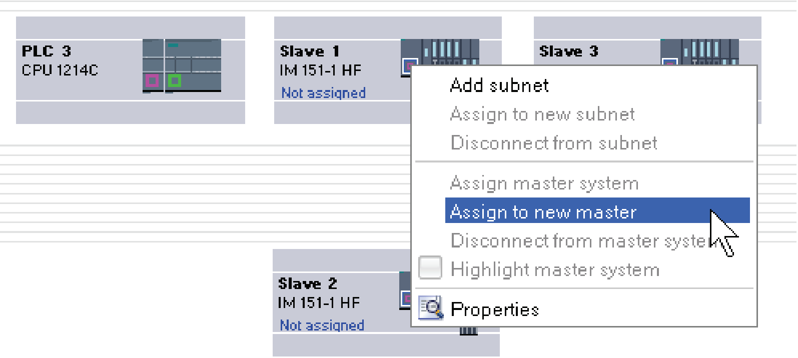
Assigning DP slaves to a DP master system

To do this, follow these steps:

1. Select an appropriate zoom factor so that you can see as many DP slaves as possible in the network view.
2. Arrange the DP slaves in a maximum of two rows.
3. Select all DP interfaces with the mouse cursor (not all devices!). This only works if you begin to drag the mouse cursor outside of the first DP slave and release the mouse button at the last DP slave (selection with the lasso).



4. Select the shortcut menu "Assign to new master" and select the corresponding DP interface for the DP master in the subsequent dialog.



5. The DP slaves are automatically networked with the DP master and combine with it to form a DP master system.

Note

When a DP master system is highlighted, you can double-click on a DP slave in the hardware catalog and thereby quickly add additional DP slaves. This will result in the DP slave being added to the highlighted DP master system automatically.

Creating a DP master system

Introduction

To create a DP master system, you need to have one DP master and at least one DP slave. As soon as you connect a DP master to a DP slave, a master-slave link is established.

DP master

You can use any of the following devices as a DP master:

- CM 1243-5

Requirements

- You must be in the network view.
- The hardware catalog is open.

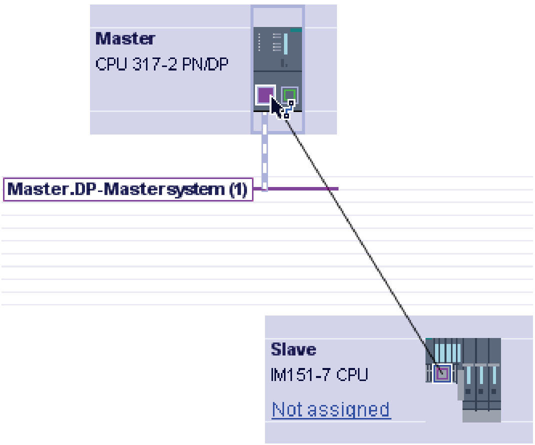
Procedure

To create a DP master system, follow these steps:

1. Select a DP master from the hardware catalog.
2. Pull the DP master onto the free area within the network view.
3. Right-click on the DP master's DP interface.
4. Select "Create master system" from the shortcut menu.

   A DP master system with one DP master will be created as a single node.

If you connect a DP slave's DP interface to that of the DP master, the DP slave will be added to the master system.



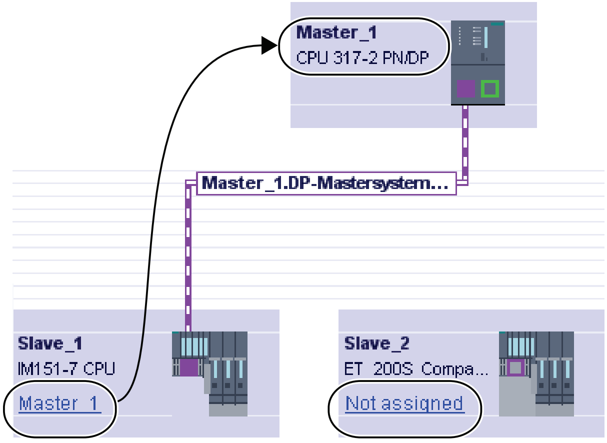
Assuming that you have already placed both a DP master and a DP slave within the network view, you can drag-and-drop to connect the two and thereby create a DP master system. To do so, follow these steps:

1. Click on the DP interface of either the DP master or DP slave.
2. Hold down the mouse button and draw a connecting line between this DP interface and that of the desired communications partner.

This will create a subnet with one DP master system between the DP master and DP slave.

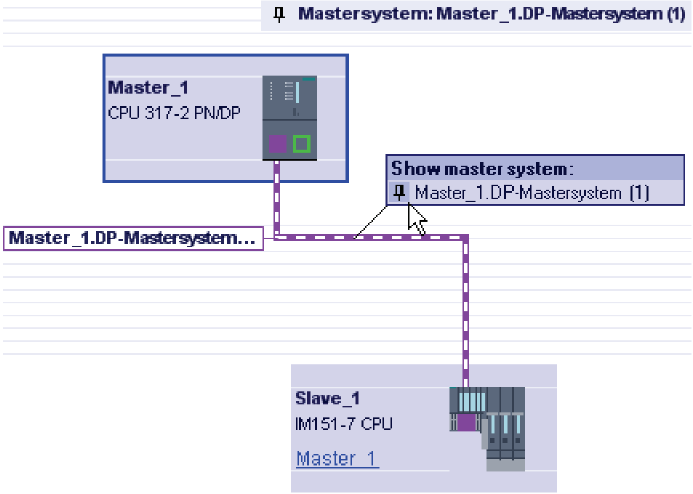
DP master display on the DP slave

When you connect a DP slave to a DP master, the name of the DP master is displayed on the DP slave as a hyperlink. Click the hyperlink to select the associated DP master.



Highlighting applied to the DP master system

When you create a new DP master system, highlighting will be applied to it. This enables you to identify quickly which devices belong to the DP master system. You can also highlight a DP master system yourself by moving the mouse pointer over a subnet. This will result in the names of the available DP master systems being displayed. Click the required DP master system to highlight it.


Mastersystem: Master_1.DP-Mastersystem (1)
Master_1
CPU 317-2 PN/DP
Show master system:
Master_1.DP-Mastersystem (1)
Master_1.DP-Mastersystem...
Slave_1
IM151-7 CPU
Master_1

There are various ways of removing the highlighting from a DP master system:

- Highlight a different master system.
- Click on the drawing pin with the name of the master system in the top right-hand corner of the network view.

Editing DP master systems and interfaces

Introduction

Once you have created a DP master system, you also have the option of disconnecting the DP master system and its components. This can result in subnets with DP slaves but no DP master.

Generally, there is no need to edit the interfaces of a DP master.

You can change the name and number on the DP master system.

Disconnection from DP master systems

If you have configured either a CPU with an integrated PROFIBUS DP interface or a PROFIBUS CP (that can be configured as an intelligent DP slave) as a DP master with master system, then you will be able to disconnect the DP master from the DP master system. After this, the device will no longer be connected to the DP master system.

Disconnecting the subnet for a DP master effectively eliminates the master system in the sense that it is no longer assigned to a DP master. However, the individual DP slaves remain interconnected via the subnet and any direct data exchange that has been configured remains unaffected.

If you delete the DP slaves or detach them from the master system, the master system is retained on the DP master.

Requirements

- You must be in the network view.
- There has to be a DP master system with one DP master and at least one DP slave.

Disconnecting the DP master from the DP master system

To disconnect the DP master system, proceed as follows:

1. Right-click on the DP master's DP interface.
2. Select "Disconnect from master system" from the shortcut menu.

The selected DP master will be disconnected from the DP master system. A subnet with the DP slaves will be retained.

Adding a DP master to the DP master system

To reassign a DP master to a subnet, proceed as follows:

1. Right-click on the DP master's DP interface.
2. Select "Create master system" from the shortcut menu.
3. Draw connecting lines between the new DP master system and the DP interfaces of the DP slaves.

The DP master combines with the DP slaves to recreate a DP master system.

Editing the properties of a DP master system

To edit the properties of a DP master system, proceed as follows:

1. Move the mouse pointer over a subnet with a DP master system.
2. A message will appear displaying the available DP master systems. Click the one you want to edit. The DP master system will now be color-highlighted.
3. Click on the highlighted DP master system.
4. In the inspector window, edit the DP master system attributes under "Properties > General".

Note

If you click a subnet when no DP master system is highlighted, you will be able to edit the properties of the entire subnet under "Properties" in the inspector window.

Adding DP slaves to the master system and configuring them

In the network view, you can add various DP slaves from the hardware catalog directly by using the drag-and-drop function or by double-clicking.

DP slaves

For configuration purposes, DP slaves are broken down into the following categories:

- Compact DP slaves
  (Modules with integrated digital/analog inputs and outputs, for example, ET 200L)
- Modular DP slaves
  (interface modules with S5 or S7 modules assigned, for example ET 200M)
- Intelligent DP slaves (I slaves)
  (CM 1242-5 or ET 200S with IM 151-7 CPU)

Rules

- Your DP master system should only contain one DP master, but it may contain one or or more DP slaves.
- You may only configure as many DP slaves in a DP master system as are permitted for the specific DP master.

Note

When configuring the DP master system, remember to observe the DP master technical data (max. number of nodes, max. number of slots, max. quantity of user data). User data restrictions may possibly prevent you from being able to use the maximum number of nodes that is theoretically possible.

Requirements

- You must be in the network view.
- A DP master system must have been created.

Adding a DP slave to the DP master system

To add a DP slave from the hardware catalog to the DP master system, follow these steps:

1. Select a DP slave from the hardware catalog.
2. Drag-and-drop the DP slave from the hardware catalog into the network view.
3. Draw a connecting line between the DP master's DP interface or a highlighted DP master system and the DP interface of the new DP slave.

A DP master system will be created and the DP slave will be connected to the DP master automatically.

Note

When a DP master system is highlighted, you can double-click the required DP slave in the hardware catalog. This will result in the DP slave being added to the highlighted DP master system automatically.

Disconnecting a DP slave from the DP master system

To disconnect a DP slave from the DP master system, follow these steps:

1. In the network view, right-click on the DP slave's DP interface.
2. From the shortcut menu, select the method for disconnecting from the DP master system:
   - "Disconnect from subnet": The PROFIBUS connection is broken and the device is no longer connected to the DP master system or a subnet.
   - "Disconnect from master system": The DP slave remains connected to the subnet, but is no longer assigned to the DP master system as a DP slave.

The selected DP slave will be disconnected from the DP master system.

Assigning a DP slave to a new DP master system

To assign an existing DP slave to a new DP master system, follow these steps:

1. Right-click on the DP slave's DP interface.
2. From the shortcut menu, select "Assign to new master".

   It does not matter whether the DP slave concerned is already assigned to another DP master system.
3. From the selection list, select the DP master whose DP master system is to have the DP slave connected to it.

   The selected DP slave is now assigned to a new DP master system.

The "Assign to new subnet" function works in a similar way in that it allows you to connect a DP slave to a new subnet. However, in this case, the DP slave will not be connected to an existing DP master system.

Configuring a DP slave

To configure a DP slave, follow these steps:

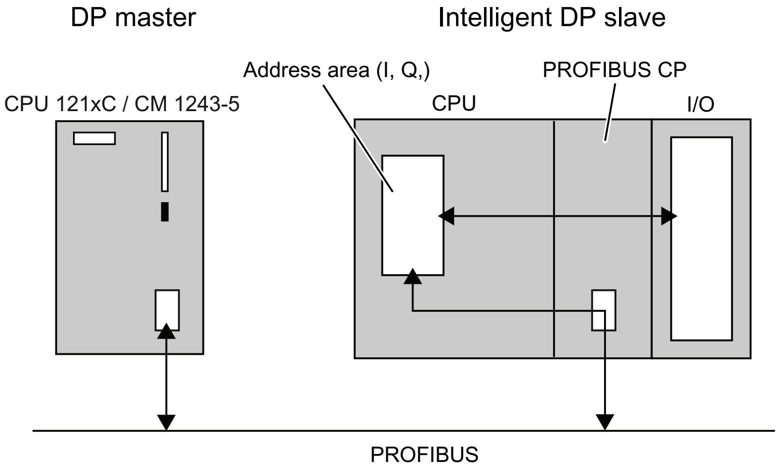
1. Switch to the DP slave's device view.
2. Select the module you want.
3. Configure the DP slave in the Inspector window.

Configuring intelligent DP slaves

Adding an I slave to a DP master system

Introduction

One of the characteristics of an intelligent DP slave (I slave) is that the DP master is not provided with I/O data directly by a real I/O, but by a preprocessing CPU. Together with the CP, this preprocessing CPU forms the I slave.



Difference: DP slave v. intelligent DP slave

In the case of a DP slave, the DP master accesses the distributed I/O directly.

In the case of an intelligent DP slave, the DP master actually accesses a transfer area in the I/O address space of the preprocessing CPU instead of accessing the connected I/O of the intelligent DP slave. The user program running on the preprocessing CPU is responsible for ensuring data exchange between the address area and I/O.

Note

The I/O areas configured for data exchange between the DP master and DP slaves must not be used by I/O modules.

Applications

Configurations involving intelligent DP slaves:

- I slave <> DP master data exchange
- Direct DP slave > I slave data exchange

Procedure

To add an I slave to a DP master system, follow these steps:

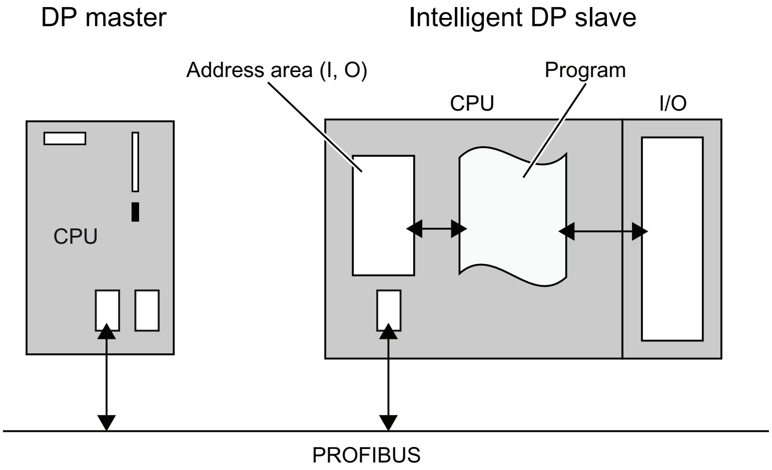
1. Drag two devices with PROFIBUS DP interface for configuration as DP master and I slave from the hardware catalog to the network view.
2. Draw a connecting line between the DP interfaces of both devices. This way you connect the I slave with a DP master in a DP master system.

   Result: You have now set up a DP master system with one DP master and one I slave.

Configuring access to I slave data

Data access

The following generally applies to I slaves: The address for the data transfer area and the address for the module on the I slave are different. This means that the start address occupied by a module can no longer be used for the transfer memory. If the higher-level DP master is to access the data of an I slave module, you will need to program this form of data exchange between the module and transfer area in the I slave user program.



Configuring the transfer area for data with DP slaves of the S7-1200

With CM 1242-5, the transfer area for the cyclic PROFIBUS data exchange is configured as transfer area in the parameter group "PROFIBUS interface > Mode > I Slave Communication".

Direct data access from CPU to CPU

Direct data access from CPU to CPU via PROFIBUS is supported by the S7-1200 PROFIBUS CMs only via the PUT/GET services.

Configuring DP slaves as distributed I/O devices

Configuring an ET 200S

Slot rules for configuring an ET 200S

The following rules apply when configuring an ET 200S:

- Do not leave any gaps when inserting the ET 200S modules.
- Slot 1: only for PM-E or PM-D Power Modules.
- To the left of an Electronics Module (EM): an EM or a Power Module (PM-E or PM-D) only.
- To the left of Motor Starter (MS): an MS, a PM-D, PM-D Fx (1..x..4) Power Module or a PM-X Power Module only.
- To the left of a PM-X: a motor starter or a PM-D only
- Up to 63 modules and one IM Interface Module are permitted

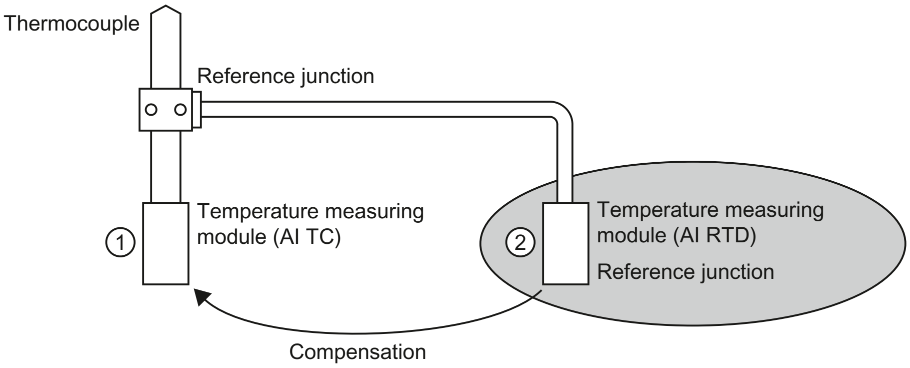
Note

Remember to ensure that the correct PM-E and EM voltage ranges are assigned.

Configuring reference junctions

A reference junction is the connection of a thermocouple to a supply line (generally in the terminal box). The voltage that occurs due to the effects of temperature falsifies the temperature value measured by the module.

On the ET 200S, one channel of the AI RTD module can be programmed as a reference junction. Other AI TC modules can correct their measured values using the temperature at the reference junction as measured by this module.


Thermocouple
Reference junction
Temperature measuring module (AI TC)
①
②
Temperature measuring module (AI RTD)
Reference junction
Compensation

① Configuring the AI TC:
→ Selection of the reference junction used

② Configuring of the AI RTD:
→ Activation of the reference junction
→ Specifying the slot and channel of the AI RTD

Special characteristics to be noted when assigning parameters for reference junctions

The process of assigning parameters for reference junctions will be explained based on the example of a resistance thermometer with a Pt 100 climatic range that is used for measuring the reference junction temperature.

To assign parameters for the reference junction, follow these steps:

1. In the ET 200S device view, insert an analog electronics module, for example a 2AI RTD HF.
2. Select the module on the rack.
3. Under "Properties > Inputs" in the inspector window, set a channel for the reference junctions function to the "RTD-4L Pt 100 climatic range" measuring range.
4. Select the ET 200S.
5. Under "Properties > Module parameters > Reference junctions" in the inspector window, select the "Reference junction" check box and specify the slot and channel number of the relevant RTD module.
6. Insert the analog electronics module for measuring the temperature using a thermocouple (TC module) and parameterize it with the reference junction number of the RTD module.

Additional information

For additional information on the various types and uses of ET 200S modules, please refer to the operating instructions and the manual titled "ET 200S Distributed I/O System".

For additional information on analog value processing, please see the documentation for the ET 200S distributed I/O system.

Packing addresses

Introduction

DP slaves and I/O devices from the ET 200S family are configured in the same way as other modular DP slaves and I/O devices. As well as supporting all the standard modular DP slave and I/O device functions, the ET 200S also offers the "Pack addresses" function:

When digital electronics modules requiring an address space of 2 or 4 bits are inserted into the device view, they will initially be spread over a total area of 1 byte. However, the address area actually occupied can be compressed after configuration using the "Pack addresses" function.

	Initial state	After "Pack addresses"
Module	I address	I address
2DI (2-bit)	Byte 10	10.0...10.1
4DI (4-bit)	Byte 11	10.2...10.5

Requirements

- You are in the device view.
- An ET 200S, for example an IM 151-1, must be present.
- A pair of digital electronics modules, for example 2DI AC120V ST, must be inserted into the slots.

Packing addresses

To pack addressed, follow these steps:

1. Select the electronics modules whose addresses are to be packed. The following options are available for selecting multiple electronics modules:
   – Press and hold down <Shift> or <Ctrl> while clicking the relevant electronics modules.
   – Click off the rack and select the required electronics modules by drawing round them with the mouse.
2. Click "Pack addresses" in the shortcut menu for the selected electronics modules.

The address areas for inputs, outputs and motor starters are packed separately. The packed addresses will be displayed in the I address and Q address columns of the device overview.

How packed addresses are generated and structured

If you make use of the "Pack addresses" function, the addresses of the selected electronics modules will be packed in accordance with the following rules:

- The start of the address area is determined by the lowest address of the selected electronics modules: X.0
- If the bit address is not "0", then the next (free) byte address as of which the selected area can be compacted will be selected automatically: (X+n).0
- If no coherent area exists, the addresses will be automatically packed into any available address gaps.

Electronics modules with packed addresses and the same byte address form a packing group.

Unpacking addresses

To unpack addressed, follow these steps:

1. Select one or more electronics modules with packed addresses.
2. Click "Unpack addresses" in the shortcut menu for the selected electronics modules.

The packing groups of the selected electronics modules will be disbanded and the packed addresses for the relevant electronics modules unpacked.

The packing group will also be disbanded and the packed addresses unpacked in the following cases: if you delete electronics modules from a packing group, move electronics modules out of a packing group or insert electronics modules on a free slot within a packing group.

The start addresses of the unpacked electronics modules will be assigned to the next available byte addresses in each case.

Special characteristics of electronics modules with packed addresses

The following special characteristics apply to electronics modules with packed addresses:

- As far as the CPU is concerned, there is no way of assigning a slot for the electronics module. Consequently, the instruction GADR_LGC (SFC 5) outputs error information W#16#8099 "Slot not configured" for the actual slot of the electronic module.
- The instruction LGC_GADR (SFC 49) and SZL-ID W#16#xy91 "module status information" cannot be evaluated for an electronics module.
- The electronics module receives an additional diagnostics address via the DPV1 function, because otherwise the packed addresses would prevent interrupts from being assigned as far as the CPU is concerned.
- The "Insert/remove interrupt" is not possible. This is because the "Pack addresses" and "Insert/remove interrupt" functions are mutually exclusive.

Configuring an ET 200S with option handling

You can use the option handling function to prepare the ET 200S for future expansions (options). This section describes option handling with reserve modules.

You do this by assembling, wiring, configuring, and programming the maximum configuration envisaged for the ET 200S and by using cost-effective reserve modules (138-4AA00 or 138-4AA10) during assembly until it becomes time to replace them with the necessary electronics modules.

Note

The ET 200S can be completely factory-wired with the master cabling, as no connection exists between a reserve module and the terminals of the terminal module (and, in turn, to the process).

Requirements

- ET 200S interface module
  - IM 151-1 STANDARD (6ES7 151-1AA03-0AB0 or higher)
  - IM 151-1 FO STANDARD (6ES7 151-1AB02-0AB0 or higher)
- Power module with option handling
  - PM-E DC24..48V
  - PM-E DC24..48V/AC24..230V

Procedure

To activate option handling, follow these steps:

1. Select the IM 151-1 in the device view and enable it in "Option handling" check box under "Properties > General > Option handling" in the inspector window.
2. Select the numbered check boxes for the slots that are initially to accommodate the reserve modules prior to the future electronics modules.
3. Select the power module in the device view and enable it in the "Option handling" check box under "Properties > Addresses" in the inspector window. Reserve the necessary address space for the control and check-back interface in the process image output (PIQ) and process image input (PII).

The assembled reserve modules can be replaced with the configured modules at a later date without having to modify the configuration.

Note

The addresses for these interfaces are reserved as soon as you activate option handling on the power module. The "Option handling" function must also be activated on the DP slave (IM 151-1 STANDARD Interface Module). If it is not activated, the addresses reserved for the control and check-back interface will be released again.

Note that activating and deactivating the option handling function repeatedly can change the address of the control and check-back interface.

Option handling may be activated for one PM-E DC24..48V or one PM-E DC24..48V/AC24..230V Power Module only.

Additional information

For additional information on the assignment and significance of bytes within the process image, option handling with PROFIBUS and the use of reserve modules, please refer to the documentation for the ET 200S distributed I/O system.

How option handling works during startup

If "Startup when expected/actual config. differ" is not available, the ET 200S will still start up if a standby module is inserted instead of the configured electronics module and option handling has been activated for the slot concerned.

How option handling works during operation

During operation, the option handling mode varies in accordance with the following:

- Option handling enabled for a slot:
  Either the standby module (option) or the configured electronics module can be plugged into this slot. If any other kind of module is present on this slot, diagnostics will be signaled (no module/incorrect module).
- Option handling disabled for a slot:
  Only the configured electronics module can be plugged into this slot. If any other kind of module is present, diagnostics will be signaled (no module/incorrect module).

Standby module substitute values

- Substitute value for digital inputs: 0
- Substitute value for analog inputs: 0x7FFF

Control and evaluation in the user program

The ET 200S is equipped with a control and feedback interface for the "Option handling" function.

The control interface is located in the process image output (PIQ). Each bit in this address area controls one of the slots from 2 to 63:

- Bit value = 0: Option handling parameterized. Standby modules are permitted.
- Bit value = 1: Option handling canceled. Standby modules are not permitted on this slot.

The check-back interface is located in the process image input (PII). Each bit in this address area provides information about the modules that are actually plugged into slots 1 to 63:

- Bit value = 0: This slot has the standby module, an incorrect module or no module plugged into it.
- Bit value = 1: This slot has the configured module plugged into it.

Configuring the ET 200S in DPV1 mode

PROFIBUS DPV1 enables you to access extended PROFIBUS functions.

Requirement

- You must be in network view.
- A DP master with DPV1 functionality must be available.
- A master-slave connection must be established with PROFIBUS.

Procedure

To switch the DP slave over to DPV1, follow these steps:

1. Select the DP slave.
2. Under "Properties > Module parameters" in the Inspector window, select "DPV1" mode from the "DP interrupt mode" drop-down list.

or

1. Select the DP master.
2. In the I/O communications table, select the row with the connection between the DP master and the desired DP slave.
3. Under "Properties > Module parameters" in the Inspector window, select "DPV1" mode from the "DP interrupt mode" drop-down list.

Special characteristics

The parameters are subject to interdependencies, which are outlined below:

Parameter	DPV0 mode	DPV1 mode
Operation when target configuration does not match actual configuration	Fully available	Fully available
Diagnostics interrupt (OB 82)	Not available, not set	Fully available
Hardware interrupt (OB 40 to 47)	Not available, not set	Fully available
Insert/remove interrupt (OB 83)	Not available, not set	Only available when addresses are not packed. "Startup when target configuration does not match actual configuration" is activated automatically along with an insert/remove interrupt.

Interrupts in the case of modules with packed addresses

If the module is capable of triggering interrupts and the bit address is not equal to 0 because of packed addresses, you will need to assign a diagnostics address in the ET 200S address dialog.

The diagnostics address is required for the purpose of assigning a DPV1 interrupt to the module as an interrupt trigger. The CPU will only be able to assign an interrupt correctly and store information on the interrupt in the interrupt OB start information/in the diagnostics buffer if the module concerned has this "unpacked" address. In this context, the CPU cannot make use of "packed" addresses.

From the point of view of interrupt processing (interrupt OB), this means the module will have the assigned diagnostics address, but from the point of view of processing the input and output data in the user program, the module will have the packed addresses.

Note

When the module addresses are packed, the insert/remove interrupt for the ET 200S is unavailable.

Using GSD files

GSD revisions

What you need to know about GSD revisions

The properties of DP slaves are made available to configuration tools by means of GSD files.

Functional enhancements in the area of the distributed I/O will have an effect on the GSD specification, for example, they will require the definition of new keywords.

This results in the versioning of the specification. In the case of GSD files, the version of the specification on which a GSD file is based is called a "GSD revision".

From GSD revision 1, the GSD revision must be included as a keyword "GSD_revision" in GSD files. GSD files without this keyword will therefore be interpreted by configuration tools as GSD revision "0".

GSD files can be interpreted up to GSD revision 5. This means that DP slaves that support the following functions, for example, will be supported:

- Diagnostic alarms for interrupt blocks
- Isochronous mode and constant bus cycle time
- SYNC/FREEZE
- Clock synchronization for DP slaves

Installing the GSD file

Introduction

A GSD file (device data file) contains all the DP slave properties. If you want to configure a DP slave that does not appear in the hardware catalog, you must install the GSD file provided by the manufacturer. DP slaves installed via GSD files are displayed in the hardware catalog and can then be selected and configured.

Requirement

- The hardware and network editor is closed.
- You have access to the required GSD files in a directory on the hard disk.

Procedure

To install a GSD file, proceed as follows:

1. In the "Options" menu, select the "Install device master data files" command.
2. In the "Install device master data files" dialog box, choose the folder in which you want to save the GSD files.
3. Choose one or more files from the list of displayed GSD files.
4. Click on the "Install" button.
5. To create a log file for the installation, click on the "Save log file" button.

   Any problems during the installation can be tracked down using the log file.

You will find the new DP slave installed by means of the GSD file in a new folder in the hardware catalog.

Note

Installation of GSD file cannot be undone.

Configuring GSD-based DP slave

DP slaves that you have inserted through installation of a GSD file can be selected as usual via the hardware catalog and inserted in the network view. If you want to insert the modules of the GSD-based DP slaves, you must take into account some particular details.

Requirements

- You have installed a DP slave using a GSD file.
- You have inserted the head module in the network view in the usual manner.
- The device overview opens in the device view.
- The hardware catalog is open.

Procedure

To add the modules of a GSD-based DP slave, proceed as follows:

1. In the hardware catalog, navigate to the modules of the GSD-based DP slave.

   GSD-based DP slaves, also referred to as DP standard slaves, can be found in the "Other field devices" folder of the hardware catalog.
2. Select the desired module.
3. Use drag-and-drop to move the module to a free space in the device overview.
4. Select the module in the device overview to edit parameters.

You have now inserted the module in a free slot of the GSD-based DP slave and can edit its parameters.

Note

You can see only the GSD-based DP slave in the graphic area of the device view. The added modules of GSD-based DP slaves are only found in the device overview.

Preset configuration

For modules with an adjustable preset configuration, you can change this configuration in the inspector window under "Properties > Preset configuration".

8.1.4.3 Configurations for PROFINET IO

What you need to know about PROFINET IO

What is PROFINET IO?

PROFINET IO

PROFINET is an Ethernet-based automation standard of PROFIBUS Nutzerorganisation e.V. (PNO) which defines a manufacturer-neutral communication, automation and engineering model.

Objective

The objective of PROFINET is:

- Integrated communication via field bus and Ethernet
- Open, distributed automation
- Use of open standards

Architecture

The PROFIBUS User Organisation e.V. (PNO) has designated the following aspects for PROFINET architecture:

- Communication between controllers as components within distributed systems.
- Communication between field devices, such as I/O devices and drives.

Implementation by Siemens

The demand for "Communication between controllers as components within distributed systems" is implemented by "Component Based Automation" (CBA). Component Based Automation is used to create a distributed automation solution based on prefabricated components and partial solutions.

The demand for "Communication between field devices" is implemented by Siemens with "PROFINET IO". Just as with PROFIBUS DP, the complete configuration and programming of the components involved is possible using the Totally Integrated Automation Portal.

The following sections deal with the configuration of communication between field devices using PROFINET IO.

Overview of RT classes

RT classes in PROFINET IO

PROFINET IO is a scalable, real-time communication system based on Ethernet technology. The scalable approach is reflected in several real-time classes:

- RT: Transmission of data in prioritized, non-isochronous Ethernet frames. The required bandwidth is within the free bandwidth area for TCP/IP communication.
- IRT: Isochronous transmission of data with high stability for time-critical applications (for example, motion control). The required bandwidth is from the area of bandwidth reserved for cyclic data.

Depending on the device, not all real-time classes are supported.

Connecting existing bus systems

Connection of PROFINET and PROFIBUS

PROFINET IO and PROFIBUS DP can be connected with each other as follows:

- via Industrial Ethernet:

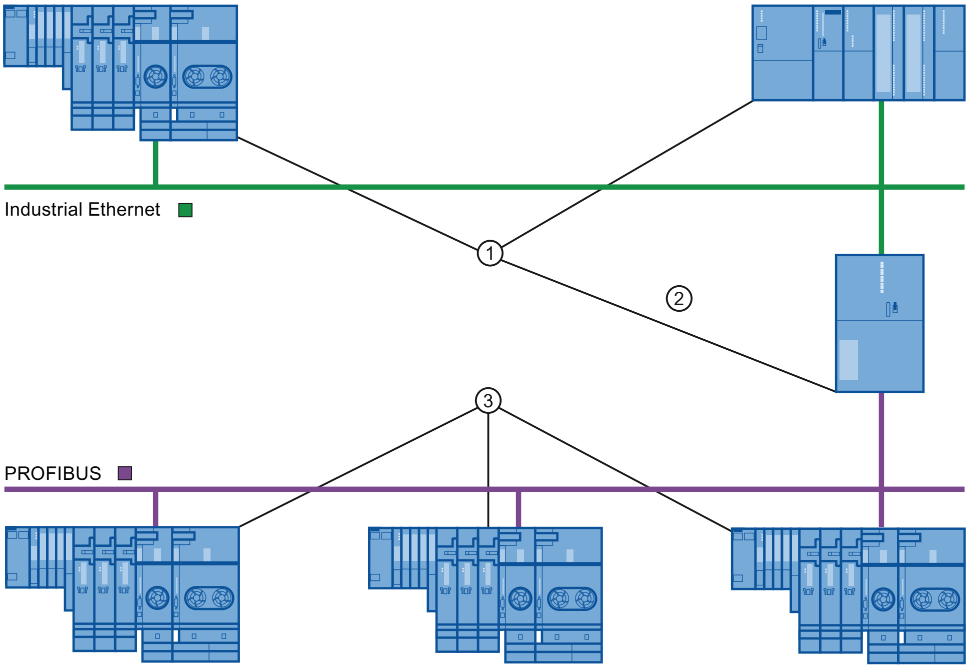
  To connect the two network types, Industrial Ethernet (control level) and PROFIBUS (cell level/field level), use e.g. the IE/PB link.

- via Industrial Wireless LAN:

  PROFIBUS devices, for example, can be connected to PROFINET IO via a wireless LAN/PB link. This allows existing PROFIBUS configurations to be integrated into PROFINET.

AS interface devices can be connected by an IE/AS-i link PN IO to the interface of a PROFINET device. This allows the existing AS-i network to be integrated into PROFINET.

The following figure shows the connection of a PROFIBUS subnet via PROFINET device with proxy functions.



① PROFINET devices
② PROFINET device with proxy functions (for example, IE/PB link)
③ PROFIBUS devices

PROFINET device with proxy functions used as proxy for a PROFIBUS device

The PROFINET device with proxy functions is the proxy for a PROFIBUS device on the Ethernet. Proxy functionality allows a PROFIBUS device that can communicate with all devices on the PROFINET and not just with its master.

Using PROFINET, existing PROFIBUS systems can easily be integrated into PROFINET communication using the proxy functions.

If, for instance, you connect a PROFIBUS device via an IE/PB link to PROFINET, the IE/PB link acts as a proxy for the PROFIBUS components to establish communication via PROFINET.

Configuration using IE/PB link PN IO

Configuration using IE/PB link PN IO

Use the IE/PB link IO to connect PROFIBUS DP configurations to PROFINET IO.

From the CPU perspective, the PROFIBUS DP slaves are connected to the same network as the IE/PB link PN IO. These slaves have the same device names and IP addresses as the IE/PB link PN IO, but different device numbers. Furthermore, each also has a specific PROFIBUS address.

In the properties of the IE/PB link, the PROFIBUS addresses of the connected DP slaves are displayed in addition to the PROFINET device numbers, as this device has two addressing schemes.

Handling device numbers and PROFIBUS addresses on the master system

During placement, the same number is assigned for the PROFINET device number and the PROFIBUS address.

In the inspector window under "General Properties > PROFINET device number", you can find an overview of the device numbers used and the PROFIBUS addresses of an IE/PB link. The device number can also be changed here. You can also set that the device number and the PROFIBUS address should or should not always be identical. If the "PROFINET device number=PROFIBUS address" is activated, you do not have to track the device number when the PROFIBUS address changes.

The PROFIBUS addresses can be changed in the properties of the PROFIBUS device.

Restrictions

The following restrictions apply to DP slaves configured as described above on the PROFIBUS subnet of an IE/PB link:

- No pluggable IE/PB link
- No pluggable DP/PA link
- No pluggable Y link
- Not CiR-compliant
- No pluggable redundant slaves
- No isochronous transmission / constant bus cycle time can be configured
- SYNC/FREEZE instructions ("DPSYC_FR") of a CPU on the the Ethernet subnet for DP slaves behind the IE/PB-Link are not supported.

Configuration using IWLAN/PN link

Maximum number of devices in a IWLAN segment

If an Ethernet subnet is set up as wireless network (IWLAN = Industrial Wireless LAN), cyclic data exchange between IO controllers and IO devices is possible via a wireless route.

On one side of the wireless route there are fixed installed access points (for example, SCALANCE W 788) and on the other side mobile stations (with, for example IWLAN/PB links with PROFIBUS devices).

If the action radius of the mobile stations is large, it may be necessary to install several access points (SCALANCE W 788). Since each access point forms a segment with its wireless range, the IWLAN is made up of a series of segments.

The mobile devices "on the one side" of the wireless link with their IWLAN/PB links can move along the segments.

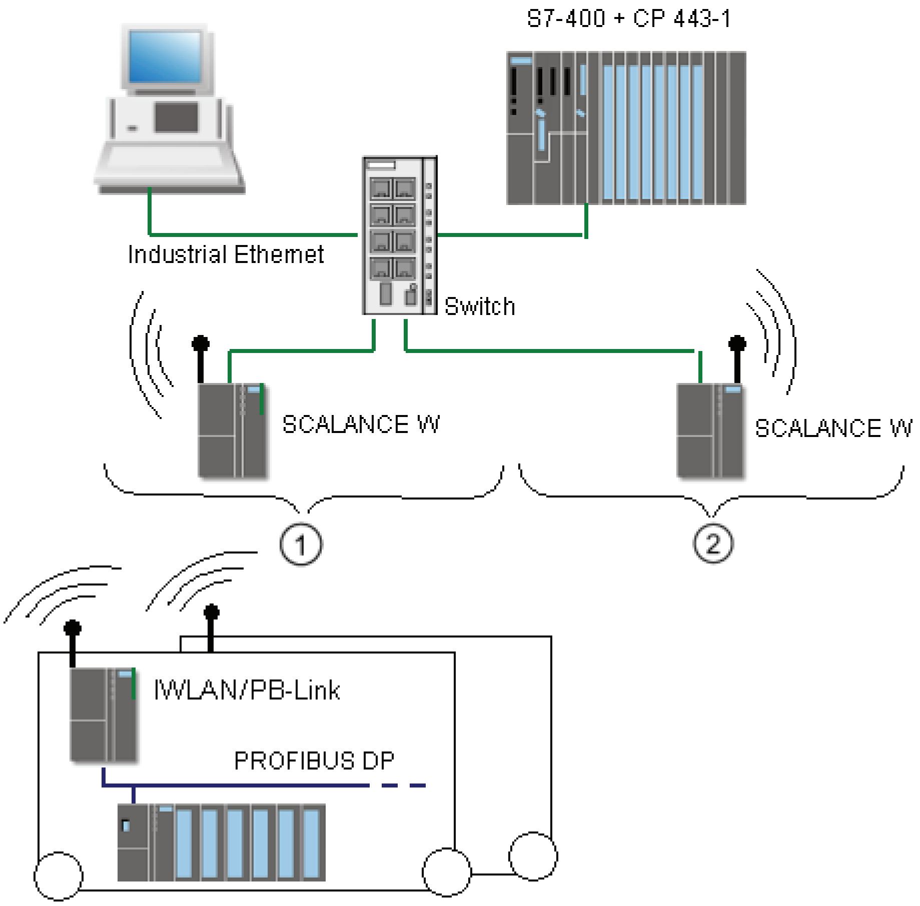
Special feature

If several IWLAN/PB links are located within a segment, they have to share the bandwidth that is available for wireless transmission. This leads to a lengthening of the update time for these devices.

Example

In the following example there are two IO devices (IWLAN/PB link) with a segment.

If no more than a maximum of two IWLAN/PB links are present in a IWLAN segment at the same time, enter a "2".



①	Segment 1
②	Segment 2

Configure PROFINET IO

Addressing PROFINET devices

Assigning addresses and names to PROFINET devices

In this chapter you will learn which address and naming conventions are valid for the PROFINET devices.

IP addresses

All PROFINET devices work with the TCP/IP protocol and therefore require an IP address for Ethernet operation.

You can set the IP addresses in the module properties. If the network is part of an existing company Ethernet network, ask your network administrator for this data.

The IP addresses of the IO devices are assigned automatically, usually at CPU startup.

Device names

Before an IO device can be addressed by an IO controller, it must have a device name. This procedure was chosen for PROFINET because names are easier to administer than complex IP addresses.

Both the IO controller as well as IO devices have a device name. The device name is automatically derived from the name configured for the device (CPU, CP or IM):

- The PROFINET device name is made up of the name of the device (for example, the CPU), the name of the interface (only with multiple PROFINET interfaces) and optionally the name of the IO system:

  <CPU name>.<Name of the interface>.<IO system name>

  You cannot change this name directly. You change the PROFINET device name indirectly, by changing the name of the affected CPU, CP or IM in the general properties of the module. This PROFINET device name is also displayed, for example, in the list of accessible devices.
- A "converted name" is generated from the PROFINET device name. This is the device name that is actually loaded into the device.

  The PROFINET device name is only converted if it does not comply to the rules of IEC 61158-6-10. You cannot change this name directly either.

Rules for the converted name

The rules for the converted name are listed in the following section. If the converted name is **not** different from the name of the module, the name of the module must comply with this rule.

- The name consists of one or more labels , which are separated by a dot [.].
- Total length of the name: 1 to 240 characters
- Length of a label: 1 to 63 characters
- A label consists of the characters [a-z0-9-]
- Labels should not start or end with the "-" character
- The first label should not start with "port-xyz" or "port-xyz-abcde" (a,b,c,d, e,x,y,z=0-9)
- The name should not have the following form: n.n.n.n (n=0-999)

Example of device names

```
device-1.machine-1.plant-1.vendor
```

If you assign this name, to a CPU for example, STEP 7 will not convert it.

Device number

In addition to the device name, a device number is also automatically assigned when an IO device is plugged in. You can change this number.

Devices in the PROFINET subnet

In a PROFINET subnet the maximum allowable number of devices is monitored during configuration.

See also

Assigning the device name and IP address (Page 495)

Retentivity of IP address parameters and device names (Page 502)

Assigning the device name and IP address

Assigning an IP address and subnet mask for an IO controller the first time

There are various options for this: During the configuration of PROFINET interface, you have to set the following:

- IP address is set in the project.
- IP address is set using a different method.

Assign an IP address	Comments
Option: Set IP address in the project: With download hardware configuration	With the downloading of the hardware configuration to the IO controller (e.g. CPU), the IP address and the device name are also downloaded.
Option: Set IP address in the project: Assign online via PROFINET interface	Connect your programming device/PC to the same network as the relevant PROFINET device. The interface of the PD/PC must be set to TCP/IP (Auto) mode. Have a list of accessible devices displayed. Select the target device via its MAC address and then assign its configured IP address before you download the hardware configuration including the configured IP address (IP address is then saved retentively).
Option: Set IP address in the project: Assign online via MPI/PROFIBUS interface	If your PROFINET device has an MPI or PROFIBUS DP interface, connect your programming device/PC directly to the PROFINET device via the MPI or PROFIBUS DP interface. The configured IP address is applied during download of hardware configuration.
Option "Use different method to obtain IP address" • Assign online • Assign via user program • Higher-level IO controller (only with I devices)	If you have selected this option in the properties of the PROFINET interface, the IP address can be assigned by the online and diagnostics editor, by the primary setup tool or by the user program ("IP_CONF" instruction).. In case of an S7-1200-CPU, make sure that access to the CPU is not protected by a password. If the CPU is write-protected, no IP address and no device name can be assigned by any other method.

Commissioning a PROFINET interface

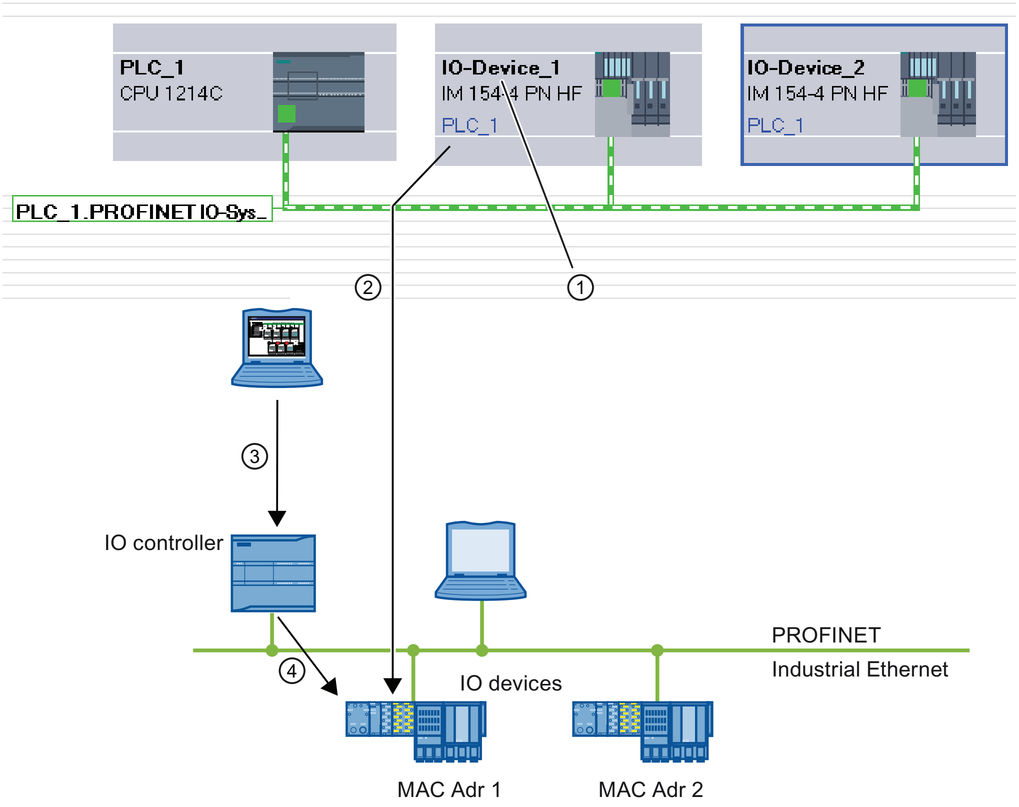
Further details of how to commission a PROFINET interface can also be found in the operating instructions for the PROFINET devices from the SIMATIC family.

Assigning device names for IO devices when the "Device replacement without removable media/PG" option is enabled

For IO devices where the "Device replacement without removable media/PG" option is activated, it is not necessary to assign the device name in the case of a device replacement.

Assigning a device name and address for an IO device (exception in the case device exchange without media change/PG)

The following graphic illustrates the process for assigning the device name and address.



① Each device receives a name; STEP 7 automatically assigns an IP address.

② STEP 7 generates a PROFINET device name from the name. This is then assigned to an IP device online (MAC address) and is written into the device.

③ The configuration is downloaded to the IO controller.

④ The IO controller assigns the appropriate IP address to the IO device with the assigned PROFINET device name during startup.

A device name is assigned to each IO device. You can change the name and IP address manually.

You have two options for downloading configured data to the PROFINET IO device:

- Offline with a Micro Memory Card:

  Place the configured data (device name: for example, turbo-3) for the IO device in the MMC in the PG/PC. Use the command "SIMATIC Card Reader > Save Device Name to Memory Card" in the "Project" menu for this.

  Then insert the MMC into the IO device. The IO device automatically adopts the configured device name.

- Online with programming device/PC:

  Connect the programming device/PC directly to the Ethernet subnetwork via the PROFINET interface.

  Select the subnet in the network view and then select the "Assign device name" shortcut command.

  In the "Assign PROFINET device name" dialog box, select the suitable PG/PC interface to connect to the Ethernet subnet.

  All configured PROFINET device names are in the top drop-down list. Select a PROFINET device name from it and select the IO device to receive this device name from the table at the bottom. You can filter the display of devices in the table according to various criteria.

  You can easily identify the device using the "Flash LED" button.

  The IO controller recognizes the IO device by its device name and automatically assigns the configured IP address to it.

IP address assignment for special IO devices

Special IO devices, for example, SCALANCE X, S7 300 CPs, support the option of assigning the IP addresses not from the IO controller during startup. In this case, the IP address is assigned in a different way. For additional information, refer to the manual of the respective PROFINET device of the SIMATIC device family.

Requirement for additional procedures when assigning IP address and device name

If the IO device, as described above, should not obtain the IP address or device name from the IO controller, proceed as follows:

1. Select device or network view.
2. Open the properties for the respective PROFINET device.
3. Select the "Use different method to obtain IP address" option or "Different method for obtaining device name" option.

Rules

If the "Different method for obtaining IP address / device name" option is used in a PROFINET device, note the following:

- The subnet part of the IP address of the IO device must match the subnet part of the IP address of the IO controller.
- The corresponding PROFINET device cannot be used as a router.

See also

Calling the name assignment function from the project tree or via the "Online" menu (Page 676)

Example of the assignment of the device name

In this example you assign device names to a PROFINET IO controller and a PROFINET IO device. To make assignment easier, the device names should also contain the names of the PROFINET IO system.

Requirements

- You must be in the network view.
- A CPU 1214C (V2.0 or higher) must be available in the network view.
- An IM 151-3PN is available.
- The PROFINET interfaces of both modules are networked.

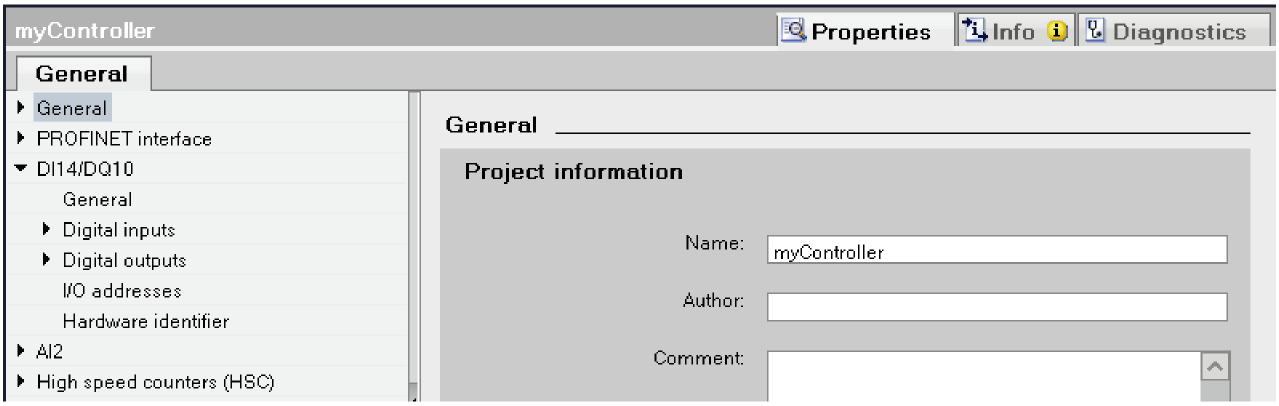
Procedure

To assign the names, follow these steps:

1. Select the CPU.

   Make sure that you have selected only the CPU and not the complete device!

2. Assign the name "myController" in the Inspector window, under "General".



myController

General

Project information

Name: myController

Author:

Comment:

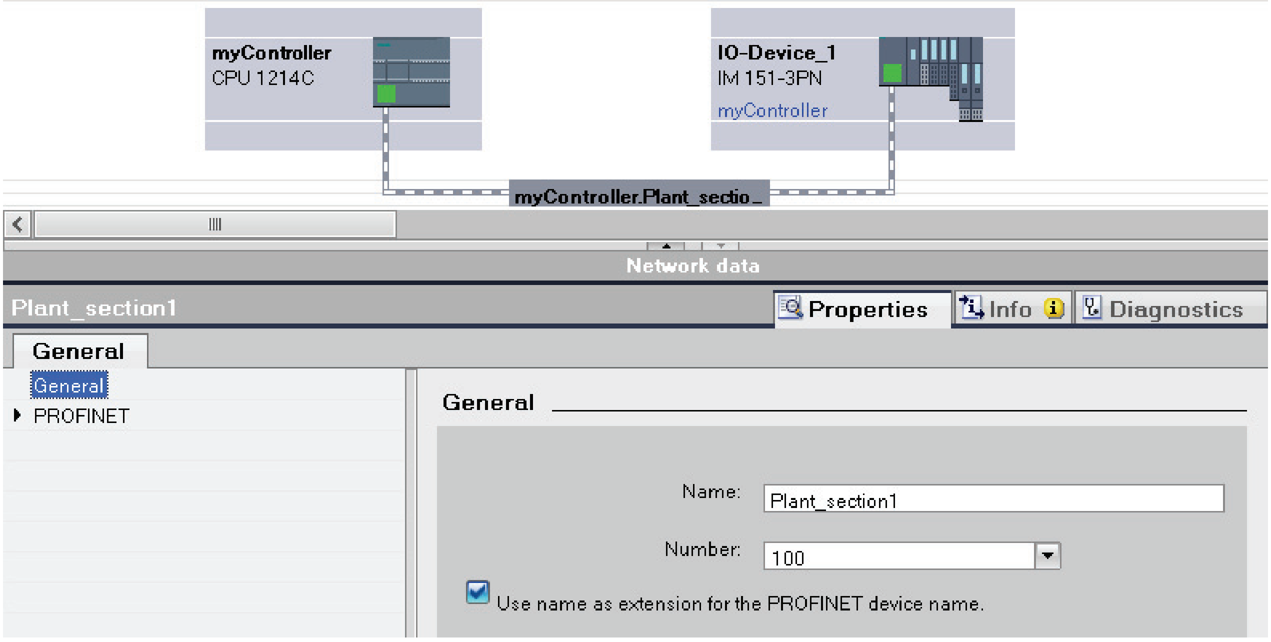
3. Select the IM.

   Make sure that you have selected only the IM and not the complete device ET200S!

4. Assign the name "Device_1" in the Inspector window, under "General".

5. Right-click on the PROFINET IO system and select the "Properties" command.

6. Assign the name "Plant_section1" to the IO system and select the check box "Use name as extension for PROFINET device names".

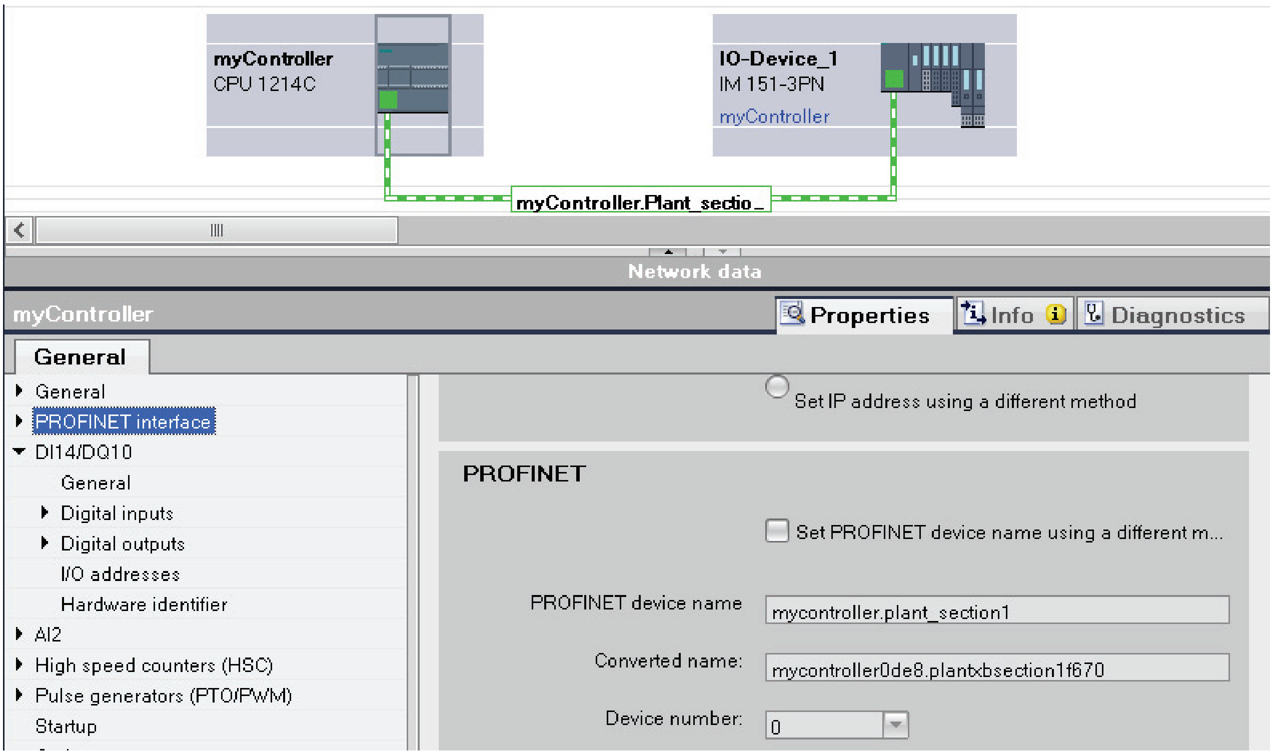


Name: Plant_section1

Number: 100

☑ Use name as extension for the PROFINET device name.

7. You can find the automatically generated PROFINET device names at the selected device in the Inspector window, at "PROFINET interface".



○ Set IP address using a different method

PROFINET

☐ Set PROFINET device name using a different m...

PROFINET device name: mycontroller.plant_section1

Converted name: mycontroller0de8.plantxbsection1f670

Device number: 0

The PROFINET device name corresponds to the name of the module (with the name of the IO system as extension) with the difference that only lower case text is used. Background: No distinction is made between upper and lower case ("case insensitive") for the storing of the name.

The converted name is displayed below. This is the name that is automatically generated from the PROFINET device name and satisfies the DNS conventions. If you work with STEP 7, you do not require this name. This name is displayed here as a check and corresponds to the name that is stored in the device. If you work with other tools that are able to record the data exchange and read the actual device names, then you find the converted names.

Other special features

For PROFINET devices with multiple PROFINET interfaces, the name of the interface is attached to the name of the module, separated by a dot.

Example:

- Name of the module: myController
- Name of the interface: Interface_1
- PROFINET device name: mycontroller.interface_1

Assign device name via memory card

Introduction

You can configure the device names of PROFINET IO devices offline. To do this, store a configured device name on a memory card and then insert the card into the appropriate IO device.

If an IO device has to be completely replaced due to a device defect, the IO controller automatically reconfigures the new device. Using the memory card, a device can be replaced without a programming device.

Requirements

- The programming device has a card reader for memory cards.
- The IO device must support the assignment of the device name via memory card.
- The station and its PROFINET IO system is configured.

Procedure

To store a device name on a memory card, follow these steps:

1. Insert the memory card into the card reader.
2. Select the IO device whose device name is to be assigned by the memory card.
3. Select the "Card reader > Save Device Name to Memory Card" command in the "Project" menu.

   If the memory card is not empty, a message will be issued informing you of this and you will have the option to delete the card.

Retentivity of IP address parameters and device names

The retentivity of IP address parameters (IP address, subnet mask, router setting) and device name depends on how the address is assigned. The non-retentive, temporary assignment means:

- IP address parameters and device name remain valid for the following time period:
  - Until the next POWER OFF
  - Until the next bootstrap
  - Until termination of the online connection (for example, after downloading the program)

    After POWER OFF / POWER ON or a permanent deletion, the CPU can only be accessed via the MAC address.
- Loading a temporary IP address also deletes the retentive, saved IP address parameter.

IP address parameters and device name not assign retentively

If the IP address parameters are not retentive, communication can no longer take place after the above described events (for example, after POWER OFF/POWER ON) that are based on the IP protocol.

IP address parameters and device name are not retentive in the following case:

- The properties of the PROFINET interface have a setting specifying that the IP address is to be obtained by another method.
- The device is a "normal" IO controller and it is specified in the user program ("IP_Conf" instruction) that the IP address parameters/device name is not to be retentive.

Non-retentive IP address parameters with an I device:

- If the device is an I device, the IP address is assigned by a higher-level IO controller and no prioritized startup is set, no IP address parameters can be assigned retentively via the online and diagnostics view. In this case the IP parameters are not retentive.

Recommendation

If possible, use the "Set IP address in project" and specify an appropriate IP address. In this case, the IP address is assigned retentively.

Resetting the IP address parameters and device names

Retentive IP address parameters and device names can be reset as follows:

- By "Reset to factory settings" (reset to delivery state with the option of also resetting the IP address parameters and the device name)
- Through a Firmware update

NOTICE
Consequences of repeated assignment of IP address parameters on top of existing IP parameters • Through the temporary assignment of IP address parameters / device names, a reset of retentively saved IP address parameters/device names occurs. • With a fixed assignment of IP address parameters/device names, previously retentively saved parameters are replaced with the newly assigned parameters.

NOTICE
Reuse of devices Execute the "Reset to factory settings" before you install a device with retentive IP address parameters / device names in different subnetworks / systems or before you place it in storage.

Creating a PROFINET IO system

A PROFINET IO system is comprised of a PROFINET IO controller and its assigned PROFINET IO devices.

To create a PROFINET IO system you require an IO controller (for example, CPU 1214C) and one or more IO devices (for example, a head module from the distributed I/O family ET 200S).

As soon as you connect an IO controller to an IO device, a controller-device link is established.

Procedure

To create a PROFINET IO system, proceed as follows:

1. Use drag-and-drop to pull an IO controller from the hardware catalog (for example, CPU 1214C) into the free area of the network view.

   The IO controller is created in the project.
2. Use drag-and-drop to move an IO device from the hardware catalog (for example, ET 200S) into the free area of the network view.
3. Click on the PROFINET interface of the IO controller or the IO device.
4. Hold down the mouse button and draw a connecting line between this selected interface and that of the partner device.

   A subnet with an IO system between the IO controller and the IO device is created.
5. If required, adapt the properties of the Ethernet subnet or the IO controller (for example, IP address) under "Properties" in the inspector window.

Handling PROFINET IO systems

Using shortcut menu commands, you can delete PROFINET IO systems, create new ones or even connect the interface to another subnet from within the network view.

An existing PROFINET configuration can thereby be corrected in the network view.

Create new PROFINET IO system for IO controller

To create a new PROFINET IO for an IO controller, proceed as follows:

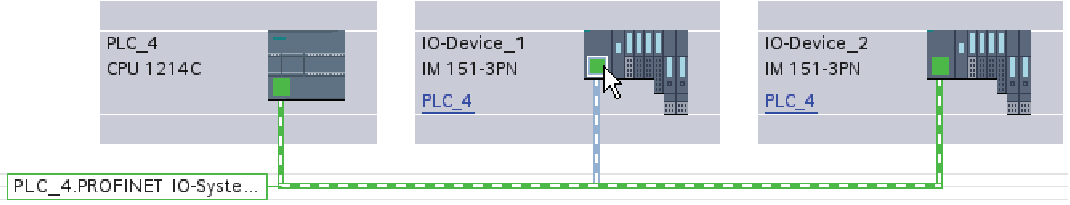
1. Make sure that no IO system is already assigned to the IO controller. If an IO system is already assigned to the IO controller, the "Assign IO system" shortcut menu command is disabled.
2. Select the PROFINET interface and then select the "Assign IO system" shortcut menu command.

A new PROFINET IO system is created at the IO controller and you can assign IO devices to this IO system.

Disconnecting PROFINET IO devices from PROFINET IO system

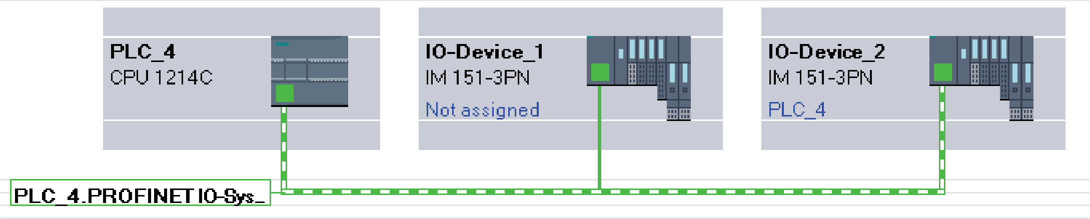
To disconnect an already networked PROFINET IO device from its PROFINET IO system, follow these steps:

1. Click on the PROFINET interface of an IO device.



2. Select the "Disconnect from IO system" shortcut menu command.

   The IO device that was assigned to this IO system is then no longer assigned to it.



You can create a new IO systems and can assign each of the non-assigned IO devices to an IO controller.

Assign PROFINET IO devices to other IO controllers

Existing PROFINET IO systems can be easily reconfigured in the network view:

1. Select the interface of an IO device and then select the shortcut menu. You have the following options here:
   - Assign a new subnet to the IO device or disconnect it from the existing subnet
   - Assign a new IO controller to the IO device
   - Assign a new IO system to the IO device or disconnect it from the existing subnet
2. To assign another IO controller to the IO device, select the "Assign to new IO controller" shortcut menu command.

   If there is no connection, a subnet is automatically created and the IO device is assigned to the IO system of the new IO controller.

Tip: Quick configuration of IO systems

If the IO system has a lot of IO devices, assign all IO devices placed by drag-and-drop operation to an IO controller on one step.

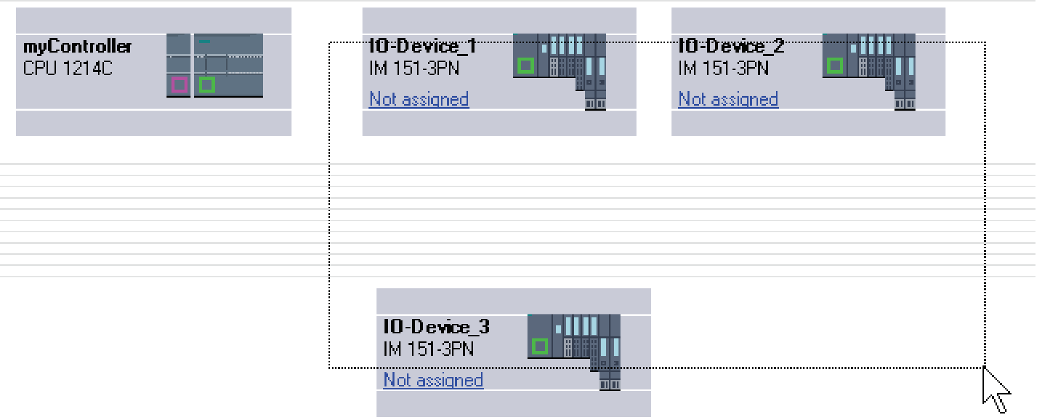
Requirements

IO controller and IO devices are placed in the network view.

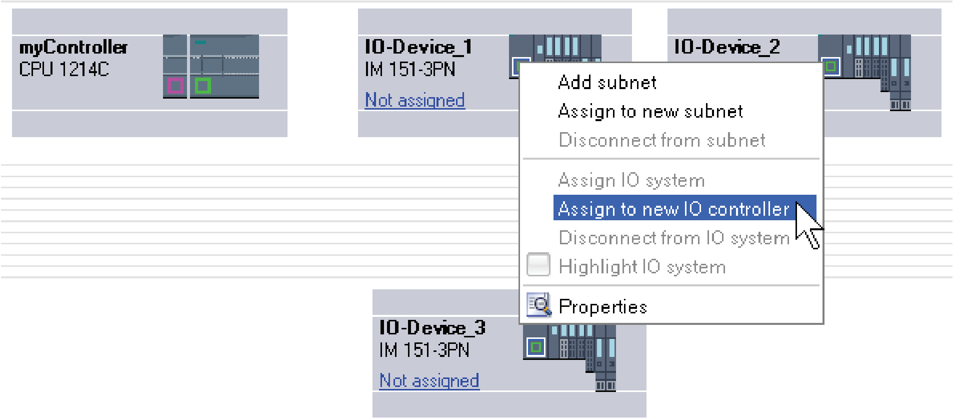
Assign IO devices to an IO system

To do this, follow these steps:

1. Select an appropriate zoom factor so that you can see as many IO devices as possible in the network view.
2. Arrange the IO devices in not more than of two rows.
3. Select all IO interfaces (not all devices) with the mouse cursor. This only works if you begin to drag the mouse cursor outside of the first IO device and release the mouse button at the last IO device (selection with the lasso).



4. Select the shortcut menu "Assign new IO controller" and select the corresponding IO interface of the IO controller in the subsequent dialog.



5. The IO devices are automatically networked with the IO controller and combine with it to form an IO system.

Note

When an IO system is highlighted, you can double-click on an IO device in the hardware catalog and thereby quickly add additional IO devices. Result: The IO device is automatically added to the highlighted IO system.

Interconnecting ports

If an IO device is assigned to an IO controller, this does not yet specify that the ports are connected to each other.

Although a port interconnection is not required to use the PROFINET functions, it does however offer the following advantages:

- A setpoint topology is assigned with the port interconnection. Based on an online-offline comparison, it is possible to conduct a setpoint-actual comparison with all devices that support this function.
- Only with IRT communication: If a port interconnection is configured, STEP 7 can determine the required bandwidth more precisely. As a rule, this leads to a higher performance.

Make sure that no invalid ring structures occur through the interconnection of ports.

Port interconnection is only advisable for devices that support the topology configuration.

Interconnecting ports in the Inspector window

To interconnect ports, follow these steps:

1. Select the PROFINET device or the PROFINET interface.
2. Navigate to the port property "Port interconnection".

   When the PROFINET interface is selected, you can find this setting in the Inspector window as follows: Properties > General > Advanced Options > Port [...] > Port Interconnection.
3. In the "Local port" section, you can find the settings at the local port. In the case of fiber-optic cable you can, for example, set the cable names here.

   In the "Partner port" section, click on the black triangle in the "Partner port" box to display and select the available partner ports.
4. If the port interconnection is a port interconnection with copper as medium and the devices support IRT communication, you can also set cable length and signal transit time.

If the PROFINET interface was not networked, it is automatically networked by this action. In the properties of the subnet you can set whether this subnet should or should not be used for the networking.

Setting the send clock

Shortest possible update interval for send clock

The send clock indicates the time between two consecutive communication cycles. It is the shortest possible transmission interval in data exchange. The update times are calculated as multiples of the send clock cycle.

Requirements to change the send clock at the PROFINET device

No IRT (Isocronous Realtime) should be configured. In detail, this means:

- No device must be configured at the IO system as a sync-slave or sync-master.
- All devices at the IO system must be unsynchronized.

Procedure

To set the send clock, proceed as follows:

1. Select the PROFINET IO controller in the device or network view.
2. Change the value for "Shortest possible update interval for send clock" in the properties of the PROFINET interface under "PROFINET Interface > Advanced options > Realtime settings > IO communication".

The send clock is valid for all PROFINET devices at the IO system. If the synchronization role is set to a value other than "Unsynchronized", you can only set the send clock in the sync domain, in other words, centrally at the PROFINET IO system.

Setting the update time

Update time

An IO device / IO controller in the PROFINET IO system is supplied with new data from the IO controller / IO device within this time period. The update time can be separately configured for each IO device and determines the time interval in which data is transmitted from the IO controller to the IO device (outputs) as well as data from the IO device to the IO controller (inputs).

STEP 7 calculates the update time automatically in the default setting for each IO device of the PROFINET IO system, taking into account the volume of data to be exchanged as well as the set send clock.

Setting the update time

If you do not want to have the update time calculated automatically, you can change the setting.

To change the update time, proceed as follows:

1. Select the PROFINET interface of the IO device in the network or device view.
2. Change the update time in the interface properties under "Advanced options > Realtime settings > IO cycle".
   - To have a suitable update time calculated automatically, select "Automatic".
   - To set the update yourself, select "Can be set" and enter the required update time in ms.
3. To ensure consistency between the send clock and the update time, activate the "Adapt update time when send clock changes" option.

   This option ensures that the update time is not set to less than the send clock.

Non-automatic setting of the send clock may result in errors if the available bandwidth is insufficient or if other limits are exceeded (for example, too many devices are configured).

Setting the watchdog time

Watchdog time

You can configure the watchdog time for PROFINET IO devices.

If the IO device is not supplied with input or output data (IO data) by the IO controller within the watchdog time, it switches to the safe state.

Do not enter the watchdog time directly, but as "Accepted number of update cycles when IO data is missing". This makes setting easier because the update time can be shorter or longer, depending on the power of the IO device or the setting.

The resulting watchdog time is automatically calculated from the "Accepted number of update cycles when IO data is missing".

Configuring the watchdog time

To specify the watchdog time, follow these steps:

1. Select the PROFINET interface of the IO device in the network or device view.
2. In the properties of the interface, navigate to "Advanced options > Realtime settings > IO cycle".
3. Select the required number of cycles from the drop-down list "Trigger watchdog after # cycles with missing IO data".

The watchdog time is subsequently calculated automatically based on the preset factor. It must not be more than 1.92 seconds.

Note

The default setting should only be changed in exceptional cases, for example, during the commissioning phase.

Calculated bandwidth for cyclic IO data

Calculated bandwidth for cyclic IO data

Adherence to the maximum available bandwidth for cyclic IO data is monitored by the system. The maximum bandwidth depends on the send clock cycle. If the send clock cycle is greater than or equal to 1 ms, the maximum bandwidth is 0.5 ms. If the send clock cycle is shorter, the maximum available bandwidth is also reduced.

The bandwidth actually required for cyclic IO data is determined by the system based on the number of configured IO devices and IO modules. Furthermore, the required bandwidth depends on the update time that is used.

In general, the calculated bandwidth increases in the following cases:

- There is a greater number of IO devices
- There is a greater number of IO modules
- The update times are shorter.

Maximum bandwidth for cyclic IO data depending on the send clock

The following table shows how the maximum available bandwidth for cyclic IO data reacts based on the send clock:

Send clock cycle	Maximum bandwidth for cyclic IO data
250 µs – 468.75 µs	<< 125 µs
500 µs – 968.75 µs	= send clock / 2
1 – 4 ms	= 500 µs

Setting port options

Setting the port options

Changing connection settings for the PROFINET IO port

You can change the network settings for the PROFINET IO port as required. By default, the settings are made automatically. In normal situations, this guarantees problem-free communication.

Possible settings for transmission medium/duplex

Depending on the selected device, you can make the following settings for "Transmission medium/duplex":

- Automatic setting

  Recommended default setting of the port. The transmission settings are automatically "negotiated" with the peer port. The "Enable autonegotiation" option is also enabled as a default, in other words, you can use cross cables or patch cables for the connection.

- TP/ITP at x Mbps full duplex (half duplex)

  Setting of the transmission rate and the full duplex/half duplex mode. The effectiveness depends on the "Enable autonegotiation" setting:

  – Autonegotiation enabled

    You can use both cross cable and patch cable.

  – Autonegotiation disabled

    Make sure that you use the correct cable (patch cable or cross cable)! The port is also monitored with this setting.

- Deactivated

  Depending on the module type, the drop down list box can contain the "- Disabled -" option. This option, for example, allows you to prevent access to an unused port for security reasons. With this setting, diagnostic events are not generated.

"Monitor" option

This option is used to activate or deactivate the port diagnostics. Examples of port diagnostics: The link status is monitored, in other words, the diagnostics are generated during link-down and the system reserve is monitored in the case of fiber optic ports.

Option "Enable autonegotiation "

The autonegotiation setting can only be changed if a concrete medium (for example, TP 100 Mbps full duplex) is selected. Whether or not a concrete medium can be set depends on the properties of the module.

If autonegotiation is disabled, this causes the port to be permanently specified, as for example, is necessary for a prioritized startup of the IO device.

You must make sure the partner port has the same settings because with this option the operating parameters of the connected network are not detected and the data transmission rate and transmission mode can accordingly not be optimally set.

Note

When a local port is connected, STEP 7 makes the setting for the partner port if the partner port supports the setting. If the partner port does not accept the setting, an error message is generated.

Wiring rules for disabled autonegotiation

Requirement

You have made the following settings for the port in question, for example, to accelerate the startup time of the IO device:

- Fixed transmission speed
- Autonegotiation incl. autocrossing disabled

This saves you the time required to negotiate the transmission rate during startup.

If you have disabled autonegotiation, you must observe the wiring rules.

Wiring rules for disabled autonegotiation

PROFINET devices have the following two types of ports:

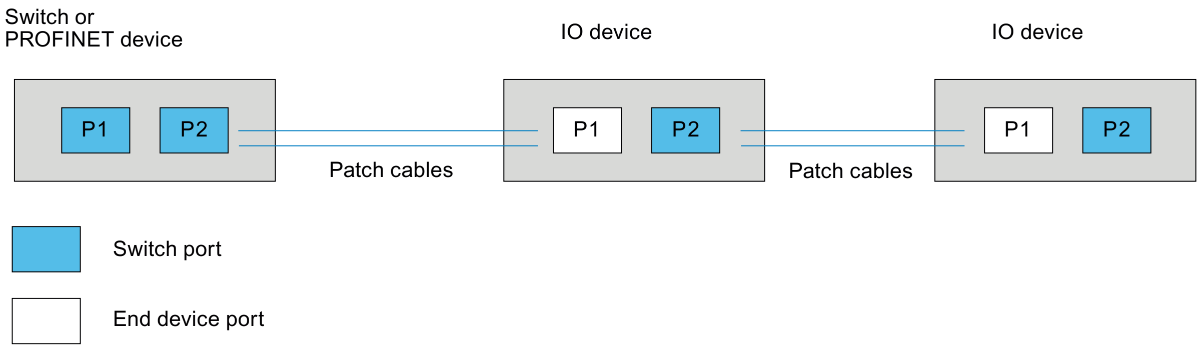
Type of port	PROFINET devices	Note
Switch port with crossed pin assignment	For IO devices: Port 2 For S7 CPUs with 2 ports: Ports 1 and 2	Crossed pin assignment means that the pin assignment for the ports for sending and receiving between the respective PROFINET devices is exchanged internally.
End device port with uncrossed pin assignment	For IO devices: Port 1 For S7 CPUs with one port: Port 1	-

Validity of the wiring rules

The cabling rules described in the following paragraph apply exclusively for the situation in which you have specified a fixed port setting.

Rules for cabling

You can connect several IO devices in line using a single cable type (patch cable). To do this, you connect port 2 of the IO device (distributed I/O) with port 1 of the next IO device. The following graphic gives an example with two IO devices.



Boundaries at the port

Requirements

To use boundaries, the respective device must have more than one port. If the PROFINET does not support boundary settings, they are not shown.

Enable boundaries

"Boundaries" are limits for transmission of certain Ethernet frames. The following boundaries can be set at a port:

- "End of discovery of accessible devices"

  No forwarding of DCP frames to identify accessible devices. Devices downstream from this port cannot be reached by the project tree under "Accessible devices". Devices downstream from this port cannot be reached by the CPU.
- "End of topology discovery"

  LLDP frames (Link Layer Discovery Protocol) are not forwarded for topology detection.
- "End of sync domain"

  No forwarding of sync frames transmitted to synchronize nodes within a sync domain.

  If you operate, for example, a PROFINET device with more than two ports in a ring, you should prevent the sync frame from being fed into the ring by setting a sync boundary (at the ports not inside the ring).

  Additional example: If you want to use several sync domains, configure a sync domain boundary for the port connected to a PROFINET device from the other sync domain.

Restrictions

The following restrictions must be observed:

- The individual check boxes can only be used if the port supports the function in question.
- If a partner port has been determined for the port, the following check boxes cannot be used:
  - "End of discovery of accessible devices"
  - "End of topology discovery"
- If autonegotiation is disabled, none of the check boxes can be used.

Enabling device replacement without exchangeable medium

Replacing an IO device without exchangeable medium

Replacement of IO devices is frequently required in automation systems. The IO devices are generally assigned a device name by either inserting an exchangeable medium or via the programming device. The CPU identifies the IO device by using these device names.

Replacing an IO device can be done without inserting an exchangeable medium (e.g. memory card) or without the programming device, under certain circumstances. For this purpose the Ethernet mechanism analyzes the relationship between the individual IO devices and the IO controller. From these relationships which are stored in the IO controller, the IO controller recognizes which IO device was replaced and assigns a device name to the new device.

Requirements

- A port interconnection is already configured.
- The affected IO devices in the automation system must support device replacement without exchangeable medium.

  If the individual IO devices in the automation system do not support device replacement without exchangeable medium, a corresponding message is output for the IO device.

  Note

  Use only new IO devices as replacements or restore configured IO devices to their delivery state.

Procedure

In order to enable the replacement of an IO device without exchangeable medium, proceed as follows:

1. In the device, select the device or network view of the PROFINET interface in the corresponding IO controller.
2. In the interface properties under "Advanced settings > Interface options", select the "Allow device replacement without exchangeable medium"

See also

Components with the the device replacement without exchangeable medium function (http://support.automation.siemens.com/WW/view/en/36752540)

Using GSDML files

GSD files for IO devices

Basic information on GSD files of IO devices

The properties of PROFINET IO devices are not stored in a keyword-based text file (as for PROFIBUS DP slaves), but in an XML file whose structure and rules are determined by a GSDML scheme.

The language used to describe the GSD files is GSDML (Generic Station Description Markup Language). It is determined by the GSDML scheme. The following sections therefore refer to GSDML files.

A GSDML scheme contains validation rules that allow it, for example, to check the syntax of a GSDML file. GSDML schemes (as scheme files) are acquired by IO device manufacturers from PROFIBUS International.

Functional enhancements in the area of PROFINET IO will have an effect on the GSDML specification and the corresponding scheme. A new version of the specification and of the scheme is created by the functional enhancement.

Names of GSDML files for IO devices

The name of a GSDML file for IO devices can, for example, be as follows:

"GSDML-V1.0-Siemens-ET200S-20030616.xml"

Name component	Explanation
GSDML	String at the start of each GSDML file for IO devices
V1.0	Version of the GSDML scheme
Siemens	Manufacturer
ET200S	Name of the device
20030616	Version code (date)
.xml	File extension

Versioning of GSDML files for IO devices

The version information of GSDML files is in two parts:

First, the version of the GSDML scheme is indicated. This determines the scope of language used by a GSDML file.

This is followed by the version, listed as an issue date. The version of GSDML files is incremented, for example, after troubleshooting or if a functional enhancement was added.

Functional enhancements may result in a new version of the GSDML scheme. A new version of a GSDML scheme might only be supported with restrictions.

Installing the GSDML file

Introduction

A GSDML file (device data file) contains all the IO device properties. If you want to configure an IO device that is not available in the hardware catalog, you must install the GSDML file provided by the manufacturer. IO devices installed via GSDML files are displayed in the hardware catalog and can then be selected and configured.

Requirements

- The hardware and network editor is closed.
- You have access to the required GSDML files in a directory on the hard disk.

Procedure

To install a GSDML file, follow these steps:

1. In the "Options" menu, select the "Install device master data files" command.
2. In the "Install device master data files" dialog box, choose the folder in which you want to save the GSDML files.
3. Choose one or more files from the list of displayed GSD- and GSDML files.
4. Click on the "Install" button.
5. To create a log file for the installation, click on the "Save log file" button.

   Any problems during the installation can be tracked down using the log file.

You will find the new IO devices installed by means of the GSDML files in a new folder in the hardware catalog.

Note

You cannot undo the installation of an GSDML file.

Changing the version of a GSDML file

Changing the revision of a GSD file

You can change the revision of a GSD file for an IO device:

- Only for the current IO device
- All suitable IO devices within the IO system
- All suitable IO devices within the complete project

First, all existing GSD files for the current IO device are shown. The only difference between the GSD files shown is their revision status. The currently used GSD file is highlighted.

Requirements

- The I/O data is the same for all IO devices whose revision is to be changed.
- The order number has not changed.
- The number of submodules is identical.
- The configuration data has not changed.
- There must be no module or submodule in a slot that is invalid after the new GSD file has been created.

Procedure

To change the revision of one or more IO devices, proceed as follows:

1. Select the IO device whose GSD file revision is to be changed.
2. Click on the "Change revision" button under "General> Catalog information" in the properties of the IO device.

   The "Change revision" dialog box opens.
3. Select the GSD revision you want to use in the "Available revisions" table.
4. Under "Use selected revision for", select the devices whose version are to be changed:
   - Only for the current IO device
   - For all suitable IO devices in the IO system
   - For all suitable IO devices in the project
5. Click the "Apply" button.

8.1.4.4 Bus coupling with PN/PN coupler

Application and function

Application

The PN/PN coupler is used to link two Ethernet subnets with one another and to exchange data. That way use data about input or output address areas or datasets can be used. The maximum size of the transferable input and output data is 1024 bytes. The division into input and output data is preferable, so that e.g. 800 byte input data and 200 byte output data can be configured.

As a device, the PN/PN coupler has two PROFINET interfaces, each of which is linked to one subnet.

In the configuration, two IO Devices are produced from this one PN/PN coupler which means that there is one IO Device for each station with its own subnet. The other part of PN/PN coupler in each case is known as the bus node. Once configuring is complete, the two parts are joined.

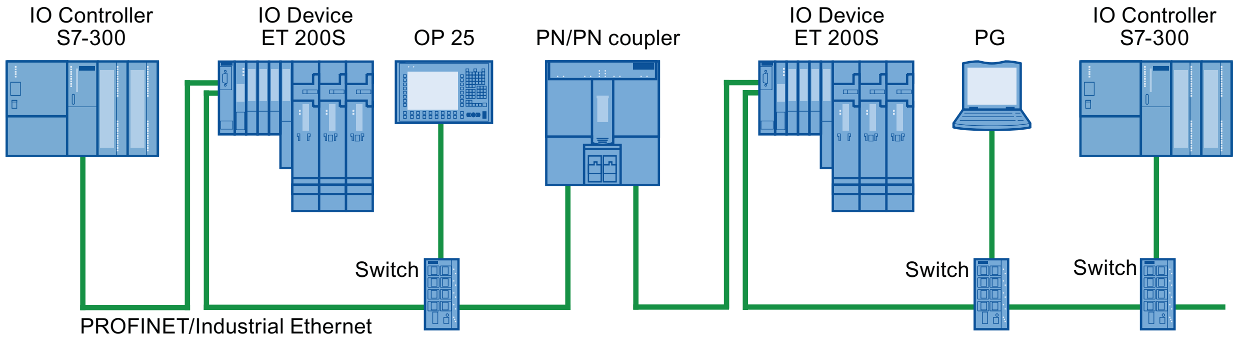


Figure 8-1 Coupling two PROFINET IO subnets with one PN/PN coupler

Additional information

For additional information on "PN/PN couplers", refer to Service & Support on the Internet (http://support.automation.siemens.com/WW/view/en/44319532).

Linking Ethernet subnets

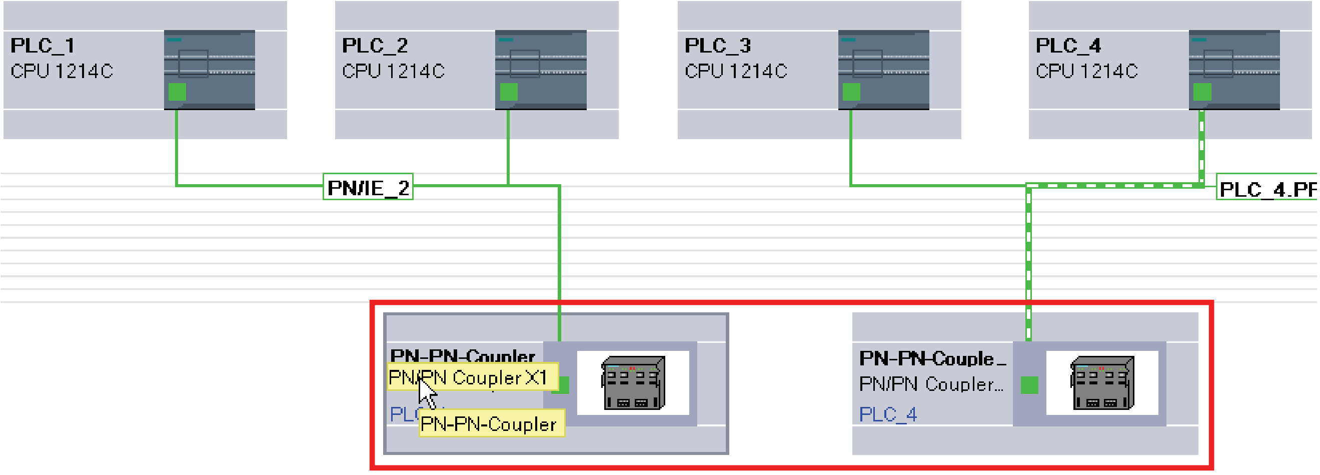
Linking Ethernet subnets with a PN/PN coupler

You can link Ethernet subnets with the standard device PN/PN coupler.

To link Ethernet subnets, follow these steps:

1. Create your Ethernet subnets.
2. Select the standard field devices in the hardware catalog. Find the PN/PN coupler as head module in the "PROFINET IO" folder.
3. In the network view, drag the two components X1 and X2 to the required version of the PN/PN coupler per drag-and-drop operation. The components form a device, but are shown separately to make handling easier.
4. Connect the Ethernet interface of the PN/PN coupler X1 to the first Ethernet subnet.
5. Connect the Ethernet interface of the PN/PN coupler X2 to the second Ethernet subnet.

   The Ethernet subnets are now linked through the two components of the PN/PN coupler.



8.1.4.5 Configurations using external tools

Integrating S7-external tools

Introduction

Tools external to STEP 7 ("Device Tools") with a special call interface (Tool Calling Interface) can be used to configure distributed devices. Such devices are also referred to as "TCI capable".

The performance range of these tools exceeds the possibilities provided within GSD configuration, for example, they can provide expanded graphical input options.

Distributed devices can be as follows:

- PROFIBUS DP slaves
- Modules within a DP slave
- PROFINET IO devices
- Modules within an IO device

Requirements

The call interface of the tool complies with the TCI specification. Parameters and commands are forwarded to the distributed device via this call interface.

Such tools have to be installed using a setup provided by the manufacturer. The "S7-PCT" (Port Configuration Tool) device tool for IO-Link modules is an exception; this is supplied with STEP 7. Special feature: After the installation, the tool is not shown in the list of installed software or in the list of software products in the project.

The GSD or GSDML file of the distributed device, which has to be configured with the device tool, has to be installed.

Starting the device tool

You can find the command to start the device tools as follows:

- Menu command "Edit > Start device tool" in the device view.
- "Start device Tool" in the shortcut menu of a TCI-enabled device.

See also

Example of a device tool (Page 522)

Example of a device tool

Introduction

The "S7-PCT" (Port Configuration Tool) device tool is installed with STEP 7.

The tool is used to assign parameters to the IO link ports of modules such as 4SI IO-Link (ET 200S) or 4IOL+8DI+4DO (ET 200eco PN).

Requirements

You have configured the corresponding DP slave or the corresponding IO device.

Starting S7-PCT

For a ET 200S with 4SI IO-Link, for example, follow these steps:

1. Select the module in the device view.
2. Select "Edit > Start device tool" from the shortcut menu.

   The tool starts and you can configure the ports.

   Alternatively, you can also start from the device view (see next section).

For a ET 200eco PN with 4IOL+8DI+4DO, follow these steps:

1. Select the module in the device view.
2. Arrange the areas in the work area in such a way that the device overview is visible (is located between device view and Inspector window).
3. Select the row with the IO link in the device overview.
4. Select "Edit > Start device tool" from the shortcut menu.

See also

Integrating S7-external tools (Page 521)

8.1.4.6 Loading a configuration

Introduction to loading a configuration

To start a device, identical configurations must be stored on the PG/PC as well as on the connected devices. Download a configuration to compare the PG/Pc and the connected devices. Configuration data can generally be downloaded in two directions:

- Download configuration from PG/PC in a device
- Download configuration from a device to the PG/PC

See also

Downloading a configuration to a device (Page 523)

Downloading a configuration to the PG/PC (Page 524)

Special features during startup (Page 530)

Downloading a configuration to a device

Downloading the hardware configuration

After you have inserted a new device in the project and configured it or if you have modified an existing hardware configuration, the next step is to load the current configuration on the device. This makes sure that the same configuration is set on the programming device/PC as well as on the physical module.

The first time you load, the entire hardware project data is loaded. When you load again later, only changes to the configuration are loaded.

You have the following options when loading the hardware configuration:

- Loading in the device or network view
- Loading in the project tree
- Loading on an accessible device

> ⚠ WARNING
>
> **Load only in STOP mode**
>
> After loading, you may experience unexpected behaviors on the machine or in the process if the parameter settings are incorrect. The CPU must be set to STOP mode for the download operation to rule out possible damage to equipment or personal injury.

Downloading a configuration to the PG/PC

Introduction

If you have connected a new device to a PG/PC but have not yet inserted the device into the project, you can transfer the complete configuration from the newly connected device to the PG/PC. The device is thereby created in the project.

A device is always downloaded via the list of accessible devices in the project tree. You can download several devices into the project by selecting them accordingly. A configuration can be downloaded several times. A new device is created on each download, even if the device has already been downloaded.

Requirements

- The original hardware configuration must be created in TIA-Portal V11.
- The opened project is in offline mode.

Scope of download

The following list presents an exact overview of the parts of the configuration that are transferred:

- Device parameters

  All set parameters of the module are transferred.
- PROFIBUS master systems and all PROFIBUS-relevant settings

  A DP master system and all connected slaves are inserted into the project. The respective settings remain unchanged. If a suitable PROFIBUS subnet has already been created, the downloaded modules are connected to the PROFIBUS interface at the existing subnet.
- PROFINET IO systems and all PROFINET-relevant settings

  The devices with IO controllers, all IO systems as well as all IO devices are transferred into the project. Settings and topologies are also transferred.

  If there already is a suitable Ethernet network in the project, the downloaded devices are integrated into the existing network.

  Relations between IO controllers and IO devices are only established within the project if both the IO controller as well as the I device are downloaded to the PG. It is unimportant whether you download the IO controller or the I devices first.
- I devices and I slaves

  Master-slave relations between the I slave and assigned DP master are only established in the project if both the master as well as the I slave are downloaded to the PG. It is unimportant whether you download the master system or the I devices and I slaves first. As soon as both devices are downloaded, the connections are also established.

- Direct data exchange

  The configuration of a direct data exchange between two devices can also be downloaded into the project. You must download both partners one after the other to do this.
- S7 connections

  S7 connections are automatically accepted as configured at one end when downloading a device configuration, even if the S7 connection was configured at both ends in the original project. Once both connection partners are downloaded, the connection is once again linked together during the next compiling procedure.
- Bus parameter

  Downloaded bus parameters initially differ from the settings in the original project after downloading a single device. The bus parameters only match those of the original project after all devices involved are downloaded and there are no additional devices on the same bus.
- An I/O module associated with the CPU

  After downloading a CPU, all other modules within the address area of the CPU are also downloaded.

8.1.5 Additional information on configurations

8.1.5.1 Functional description of S7-1200 CPUs

Operating modes

Principles of the operating modes of S7-CPUs

Introduction

Operating modes describe the behavior of the CPU. The following operating modes are possible:

- STARTUP
- RUN
- STOP

In these operating modes, the CPU can communicate via the PN/IE interface, for example.

Other operating modes

If the CPU is not ready for operation, it is in one of following two operating modes:

- Deenergized, i.e. the supply voltage is switched off.
- Defective, which means an internal error has occurred.

  If the "Defective" status is caused by a firmware error, this state is indicated by the status LEDs of the CPU (refer to the description of the CPU). To find out the cause, follow these steps:

  – Turn the power supply switch off and on again.

  – Read out the diagnostics buffer when the CPU starts up and send the data for analysis to Customer Support.

  If the CPU does not start up, replace it.

See also

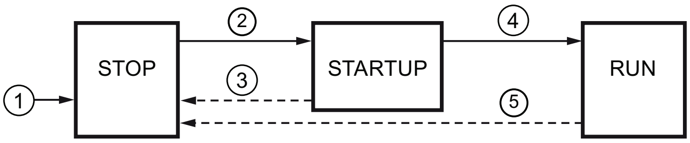
STOP mode (Page 532)

RUN mode (Page 531)

Operating mode transitions

Overview

The following figure shows the operating modes and the operating mode transitions of S7-1200 CPUs:



The following table shows the conditions under which the operating modes will change:

No.	Operating mode transition	Conditions
①	STOP	After you turn on the power supply, the CPU is in "STOP" mode. It then determines the required type of startup and changes to the next operating mode.
②	STOP → STARTUP	If the hardware configuration and the program blocks are consistent, the CPU changes to "STARTUP" mode in the following situations: • The CPU is set to "RUN" from the programming device. • After automatic triggering of a STARTUP operating mode by "POWER-ON".
③	STARTUP → STOP	The CPU returns to the "STOP" mode in the following situations: • An error is detected during startup. • The CPU is set to "STOP" from the programming device. • A STOP command is executed in the STARTUP OB.

No.	Operating mode transition	Conditions
④	STARTUP → RUN	If the STARTUP is successful, the CPU switches to "RUN".
⑤	RUN → STOP	The CPU returns to the "STOP" mode in the following situations: • An error is detected that prevents continued processing. • The CPU is set to "STOP" from the programming device. • A STOP command is executed in the user program.

"STARTUP" operating mode

Principles of the STARTUP mode

Function

After turning on the CPU, it executes a startup program before starting to execute the cyclic user program.

By suitably programming startup OBs, you can specify certain initialization variables for your cyclic program in the startup program. There is no rule in terms of the number of startup OBs. That is, you can set up one or several startup OBs in your program, or none at all.

Parameter settings for startup characteristics

You can specify whether the CPU remains in STOP mode or whether a warm restart is run. Over and above this, you can set the response during startup (RUN or previous mode) in the "Startup" group of the CPU properties.

Special characteristics

Note the following points regarding the "STARTUP" mode:

- The startup OBs are executed. All startup OBs you have programmed are executed, regardless of the selected startup mode.
- No time-based program execution can be performed.
- Interrupt controlled program execution limited to:
  - OB 82 (diagnostics interrupt)
- The outputs on the modules are disabled.
- The process image is not updated; direct I/O access to inputs is possible.

See also

Editing properties and parameters (Page 345)

Principles of the operating modes of S7-CPUs (Page 525)

Organization blocks for startup (Page 573)

Warm restart (Page 528)

Warm restart

Function

During a warm restart, all non-retentive bit memory is deleted and non-retentive DB contents are reset to the initial values from load memory. Retentive bit memory and retentive DB contents are retained.

Program execution begins at the call of the first startup OB.

Triggering a warm restart

You can trigger a "Warm restart" using a corresponding menu command on your programming device in the following situations:

- The CPU must be in "STOP" mode.
- After a memory reset
- After downloading a consistent program and a consistent hardware configuration in the "STOP" mode of the CPU.

"POWER ON" triggers a "warm restart" if you have set the following parameters for the startup response:

- Startup type "warm restart - RUN" (regardless of the CPU operating mode prior to POWER OFF).
- "Warm restart - mode prior to POWER OFF" (depending on the CPU operating mode prior to POWER OFF. The CPU must have been in RUN mode prior to this.)

See also

Retentive memory areas (Page 537)

Startup activities

Overview

The following table shows which activities the CPU performs at STARTUP:

Activities in execution sequence	At warm restart
Clear non-retentive bit memories	Yes
Clear all bit memories	No
Clear the process image output	Yes
Processing startup OBs	Yes
Update the process image input	Yes
Enable outputs after changing to "RUN" mode	Yes

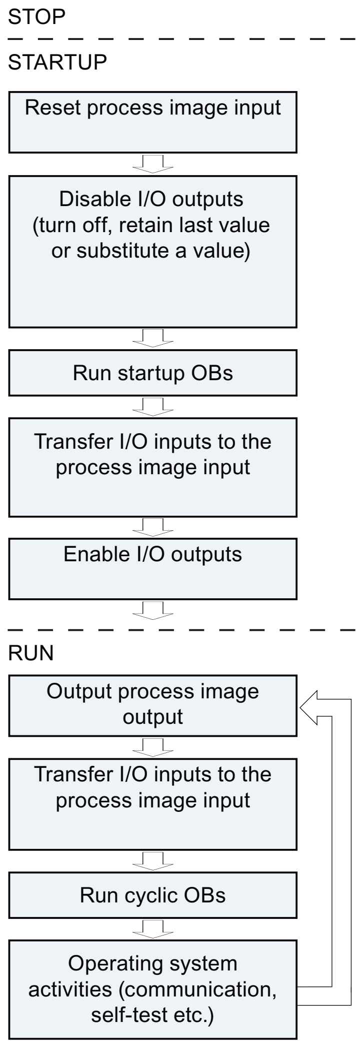
Sequence

The following figure shows the activities of the CPU in "STOP", "STARTUP", and "RUN" modes.

You can use the following measures to specify the state of the I/O outputs in the first cycle of the user program:

- Use assignable output modules to be able to output substitute values or to retain the last value.
- Set default values for outputs in startup OBs.

During the startup, all interrupt events are entered in a queue so that they can be processed later during RUN mode. In RUN mode, hardware interrupts can be processed at any time.


STOP
STARTUP
Reset process image input
Disable I/O outputs (turn off, retain last value or substitute a value)
Run startup OBs
Transfer I/O inputs to the process image input
Enable I/O outputs
RUN
Output process image output
Transfer I/O inputs to the process image input
Run cyclic OBs
Operating system activities (communication, self-test etc.)

Special features during startup

Response when expected and actual configurations do not match

The expected configuration is represented by the engineering configuration loaded on the CPU. The actual configuration is the actual configuration of the automation system.

If the expected configuration and actual configuration differ, the CPU nevertheless initially changes to RUN.

Canceling a STARTUP

If errors occur during startup, the startup is canceled and the CPU remains in "STOP" mode.

Under the following conditions, a startup will not be performed or will be canceled:

- If an invalid SD card is inserted.
- If no hardware configuration has been downloaded.

See also

Overview of the CPU properties (Page 548)

RUN mode

Function

In "RUN" mode the cyclic, time-driven, and interrupt-driven program sections execute:

- The process image output is read out.
- The process image input table is read.
- The user program is executed.

Active data exchange between S7-1200 CPUs by means of Open User Communication is only possible in "RUN" mode.

Running the user program

Once the CPU has read the inputs, the cyclic program runs from the first to the last instruction.

If you have configured a minimum cycle time, the CPU will not end the cycle until this minimum cycle time is up even if the user program is completed sooner.

A maximum cycle time is set which you can adjust according to your requirements. This ensures that the cyclic program is completed within a specified time. The system will respond with a time error if the cyclic program is not completed within this time.

Other events such as hardware and diagnostic interrupts can interrupt the cyclic program flow and prolong the cycle time.

See also

Principles of the operating modes of S7-CPUs (Page 525)

Events and OBs (Page 540)

STOP mode

Function

In "STOP" mode, the user program is not executed. All outputs are disabled or react according to the parameter settings: They provide a substitute value as set in the parameters or retain the last value output and bring the controlled process to a safe status.

The CPU checks the following points:

- Hardware, for example whether are all modules are available
- Whether the default settings for the CPU are applicable or parameter sets are present
- Whether the general conditions for the programmed startup behavior are correct

See also

Principles of the operating modes of S7-CPUs (Page 525)

Basics of a memory reset

Function

A memory reset on the CPU is possible only in STOP mode.

When memory is reset, the CPU is changed to an "initial status". This means:

- The content of the work memory and the retentive and non-retentive data are deleted.
- The load memory (code and data blocks) is then copied to work memory. As a result, the DBs no longer have current values but their initial values.
- An existing online connection between your programming device/PC and the CPU is terminated.
- The diagnostics buffer, the time, the IP address, the hardware configuration and active force jobs are retained.

Memory areas

Things you should know about memory cards

How the memory card functions

The SIMATIC Memory Card for a S7-1200 is an SD memory card preformatted by Siemens for the CPU user program.

You may only delete files and folders. If you format the memory card with Windows, for example with a commercially available card reader, you make the memory card unusable as a storage medium for a S7-CPU.

Setting the card type

You can use the memory card as a transfer card, a program card or a firmware update card.

To set the card type, insert the memory card in the programming device's card reader and select the "SIMATIC card reader" folder from the project navigation. In the properties of the selected memory card, designate the card type:

- Program

  If it is used as a program card, you can load the user program on the memory card. In this case, the internal load memory of the device is replaced by the memory card and the internal load memory is erased. The user program is then fully executable from the memory card. If the memory card with the user program is removed, there is no longer a program available.

- Transfer

  If it is used as a transfer card, you can transfer the user program from the memory card to the internal load memory of the CPU. You can then remove the memory card again.

- Firmware card

  Firmware for the S7-1200 modules can be stored on a memory card. Therefore it is possible to perform a firmware update with the help of a specifically prepared memory card. Likewise, a backup copy of firmware for a module can be stored on a memory card.

Transferring objects from the project to a memory card

When the memory card is inserted in the programming device or in an external card reader, you can transfer the following objects from the project tree to the memory card:

- Individual blocks (multiple selection possible)

  In this case a consistent transfer is available, as the dependencies of the blocks to each other is taken into account with block selection.
- PLC

  In this case, all objects relevant to processing are transferred, such as blocks and the hardware configuration on the memory card, just as with downloading.

To perform a transfer, you can move the objects by dragging and dropping, or use the command "Write to memory card" in the "Project" menu.

Transferring objects from the memory card to the project

You can transfer Individual blocks (multiple selection is possible) by dragging them to the project. A hardware configuration cannot be transferred from the memory card to the project.

Updating firmware with a memory card

You can get the latest firmware data on the internet from the Service & Support pages:

http://support.automation.siemens.com
(http://support.automation.siemens.com/WW/view/en/34143537)

Store the firmware data on the hard drive.

To store the data on the micro memory card, select the command "SIMATIC Card Reader > Create Firmware-Update Memory Card" in the "Project" menu.

Then follow the instructions in the Service & Support portal for performing a firmware update with your CPU.

Updating the firmware changes the CPU firmware status. If you have used the CPU in the project, you will have to update the CPU already configured to the CPU with the new firmware status by changing devices offline, and adapt and then load the program or configuration.

See also

Replacing a hardware component (Page 344)

Load memory

Function

Each CPU has an internal load memory. The size of this internal load memory depends on the CPU used.

This internal load memory can be replaced by using external memory cards. If there is no memory card inserted, the CPU uses the internal load memory; if a memory card is inserted, the CPU uses the memory card as load memory.

The size of the usable external load memory cannot, however, be greater than the internal load memory even if the inserted SD card has more free space.

Work memory

Function

Work memory is a non-retentive memory area for storing elements of the user program that are relevant for program execution. The user program is executed exclusively in work memory and system memory.

System memory

System memory areas

Function

System memory contains the memory elements that each CPU makes available to the user program, such as the process image and bit memory.

By using appropriate operations in your user program, you address the data directly in the relevant operand area.

The following table shows the operand areas of the system memory:

Operand area	Description	Access via units of the following size:	S7 notation
Process image output	The CPU writes the values from the process image output table to the output modules at the start of the cycle.	Output (bit)	Q
		Output byte	QB
		Output word	QW
		Output double word	QD
Process image input	The CPU reads the inputs from the input modules and saves the values to the process image input table at the start of the cycle.	Input (bit)	I
		Input byte	IB
		Input word	IW
		Input double word	ID
Bit memory	This area provides storage for intermediate results	Bit memory (bit)	M
		Memory byte	MB

Operand area	Description	Access via units of the following size:	S7 notation
	calculated in the program.	Memory word	MW
		Memory double word	MD
Data block	Data blocks store information for the program. They can either be defined so that all code blocks can access them (global DBs) or assigned to a specific FB or SFB (instance DB). Requirement: The block attribute "Optimized block access" is not enabled.	Data bit	DBX
		Data byte	DBB
		Data word	DBW
		Data double word	DBD
Local data	This area contains the temporary data of a block while the block is being processed. Requirement: The block attribute "Optimized block access" is not enabled. Recommendation: Access local data (temp) symbolically.	Local data bit	L
		Local data byte	LB
		Local data word	LW
		Local data double word	LD
I/O input area	The I/O input and output areas permit direct access to central and distributed input and output modules.	I/O input bit	<tag>:P
		I/O input byte	
		I/O input word	
		I/O input double word	
I/O output area		I/O output bit	
		I/O output byte	
		I/O output word	
		I/O output double word	

See also

Diagnostics buffer (Page 539)

Basic principles of process images (Page 537)

Access to the I/O addresses (Page 540)

Retentive memory areas

Retentive memory areas

Data loss after power failure can be avoided by marking certain data as retentive. This data is stored in a retentive memory area. A retentive memory area is an area that retains its content following a warm restart, in other words, after cycling the power when the CPU changes from STOP to RUN.

The following data can be assigned retentivity:

- Bit memory: The precise width of the memory can be defined for bit memory in the PLC tag table or in the assignment list.
- Tags of a function block (FB): You can define individual tags as retentive in the interface of an FB if you have enabled optimized block access. Retentivity settings can be defined only in the assigned instance data block if optimized block access has not been activated for the FB.
- Tags of a global data block: You can define retentivity either for individual or for all tags of a global data block depending on the settings for access.
  - Block with optimized access: retentivity can be set for each individual tag.
  - Block with standard access: The retentivity setting applies to all tags of the DB; either all tags are retentive or no tag is retentive.

See also

Warm restart (Page 528)

process image input/output

Basic principles of process images

Function

When the user program addresses the input (I) and output (O) operand areas, it does not query or change the signal states on the digital signal modules. Instead, it accesses a memory area in the system memory of the CPU. This memory area is referred to as the process image.

Advantages of the process image

Compared with direct access to input and output modules, the main advantage of accessing the process image is that the CPU has a consistent image of the process signals for the duration of one program cycle. If a signal state on an input module changes during program execution, the signal state in the process image is retained until the process image is updated again in the next cycle. The process of repeatedly scanning an input signal within a user program ensures that consistent input information is always available.

Access to the process image also requires far less time than direct access to the signal modules since the process image is located in the internal memory of the CPU.

Updating the process images

Sequence

The operating system updates the process images at cyclic intervals unless defined otherwise in your configuration. The process image input/output is updated in the following order:

1. The internal tasks of the operating system are performed.
2. The process image output (PIQ) table is written to the outputs of the module.
3. The status of inputs is read to the process image input (PII) table.
4. The user program is executed with all the blocks that are called in it.

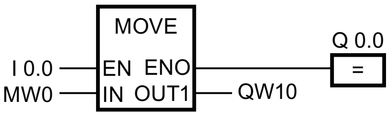
The operating system automatically controls the writing of the process image output to the outputs of the modules and the reading of the process image input.

Special characteristics

You have the option of accessing inputs or outputs directly using direct I/O access.

- If an instruction accesses an output directly and the output address is located in the process image output, the process image of the relevant output is updated.
- If an instruction accesses an output directly and the output address is **not** located in the process image output, the process image of the relevant output is **not** updated.

Example of normal I/O access by way of the process image



Update QW10 in the I/O output area with the value from MW0.

I/O access error during process image updating

If an error occurs while updating the process image (I/O access error), the CPU reacts with the default system reaction "STOP".

See also

Start address of a module (Page 539)

Access to the I/O addresses (Page 540)

Startup activities (Page 529)

Diagnostics buffer

Function

The diagnostics buffer is part of the system memory of the CPU. It contains the errors detected by the CPU or modules with diagnostics capability. It includes the following events:

- Every mode change of the CPU (for example, POWER UP, change to STOP mode, change to RUN mode)
- Every diagnostics interrupt

The diagnostics buffer of the S7-1200-CPU has a capacity of 50 entries of which the last (most recent) 10 entries are retained following power cycling.

Those entries can only be cleared by restoring the CPU to factory defaults.

You can read the content of the diagnostics buffer with the help of the Online and Diagnostics view.

See also

Basic information on the diagnostics buffer (Page 678)

I/O data area

Start address of a module

Definition

The start address is the lowest byte address of a module. It acts as the initial address of the module user data area.

Configuring module start addresses

The addresses used in the user program and the module start addresses are coordinated when the modules are configured.

In the module properties ("I/O addresses" group), you can change the start addresses that were assigned automatically after the modules were inserted.

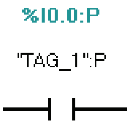
You can also make a setting that decides whether or not the addresses are located in the process image.

Access to the I/O addresses

I/O addresses

If you insert a module in the device view, its user data is located in the process image of the S7-1200-CPU (default). The CPU handles the data exchange between the module and the process image area automatically during the update of the process images.

Append the suffix ":P" to the I/O address if you want the program to access the module directly instead of using the process image.



This could be necessary, for example, during execution of a time time-sensitive program which also has to control the outputs within the same cycle.

Basics of program execution

Events and OBs

Events and OBs

The operating system of S7-1200-CPUs is based on events. There are two types of events:

- Events which can start an OB
- Events which cannot start an OB

An event which can start an OB triggers the following reaction:

- It calls the OB you possibly assigned to this event. The event is entered in a queue according to its priority if it is currently not possible to call this OB.
- The default system reaction is triggered if you did not assign an OB to this event.

An event which cannot start an OB triggers the default system reaction for the associated event class.

The user program cycle is therefore based on events, the assignment of OBs to those events, and on the code which is either contained in the OB, or called in the OB.

The following table provides an overview of the events which can start an OB, including the associated event classes and OBs. The table is sorted based on OB priority. Priority class 1 is the lowest.

Event class	OB no.	Number of OBs	Start event	OB priority
Cyclic program	1, >= 200	>= 1	Starting or end of the last cyclic OB	1
Startup	100, >= 200	>=0	STOP to RUN transition	1
Time-delay interrupt	>= 200	Max. 4	Delay time expired	3
Cyclic interrupt	>= 200		Constant bus cycle time expired	4
Hardware interrupt	>= 200	Max. 50 (more can be used with DETACH and ATTACH)	• Rising edge (max. 16) • Falling edge (max. 16)	5
			• HSC: Count value = reference value (max. 6) • HSC: Count direction changed (max. 6) • HSC: External reset (max. 6)	6
Diagnostic error interrupt	82	0 or 1	Module has detected an error	20
Time error	80	0 or 1	• Maximum cycle time exceeded • Called OB is still being executed • Queue overflow • Interrupt loss due to high interrupt load	26

The following table describes events which do not trigger an OB start, including the corresponding reaction of the operating system. The table is sorted based on event priority.

Event class	Event	Event priority	System reaction
Insert/remove central modules	Insert/remove a module	21	STOP
I/O access error during process image update	I/O access error during process image update	22	Ignore
Programming error	Programming error in a block for which you use system reactions provided by the operating system (note: the error handling routine in the block program is executed if you activated local error handling).	23	STOP
I/O access error	I/O access error in a block for which you use system reactions provided by the operating system (note: the error handling routine in the block program is executed if you activated local error handling).	24	STOP
Maximum cycle time exceeded twice	Maximum cycle time exceeded twice	27	STOP

Assignment between OBs and events

With the exception of the cyclic program and startup program and event can only be assigned to one OB. However, in certain event classes such as hardware interrupts one and the same OB can be assigned to several events.

The assignment between OBs and events is defined in the hardware configuration. Defined assignments can be changed at runtime by means of ATTACH and DETACH instructions.

OB priority and runtime behavior

S7-1200 CPUs support the priority classes 1 (lowest) to 27 (highest). An OB is assigned the priority of its start event.

OBs are always executed on a priority basis: The OBs with the highest priority are executed first. Events of the same priority are processed in order of occurrence. This means:

- Any OB with priority >= 2 will interrupt cyclic program execution.
- An OB of priority 2 to 25 cannot be interrupted by any event of priority group 2 to 25. This rule also applies to events of a priority higher than that of the currently active OB. Such events are processed later.
- A time error (priority 26) will interrupt any other OB.

OB start information

Certain OBs have start information, while others do not. This is explained in greater detail in the description of the relevant OB.

See also

Event-based program execution (Page 542)

Event-based program execution

OB priority and runtime behavior

S7-1200-CPUs support the priority classes 1 (lowest) to 27 (highest). An OB is assigned the priority of its start event.

Interrupt OBs can only be interrupted by time error interrupts. This rule also applies to events of a priority higher than that of the currently active OB. That is, only one interrupt OB can be active, with exception of the time error interrupt OB.

Any further event of generated while an interrupt OB is being executed is added to a queue in accordance with its priority. Start events within a queue are processed later based on the chronological order of their occurrence.

Program execution on the CPU

Cyclic OBs are interrupted by interrupt OBs.

Interrupt OBs can only be interrupted by time error interrupt OBs.

The following figure shows the basic sequence:

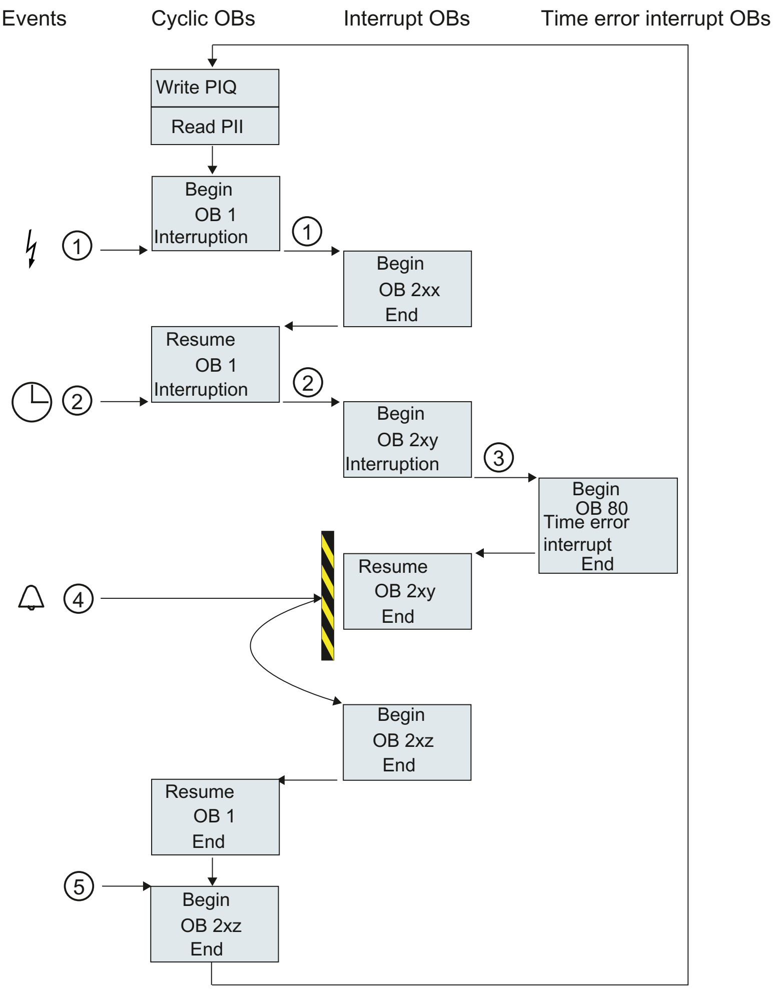


Figure 8-2 Program sequence

Description of program execution

① and ② An event (e.g. a hardware interrupt) calls its associated OB.

A called OB is executed without interruption, including all of its nested blocks. Execution of the cyclic OB is resumed on completion of interrupt processing, provided the queue does not contain any events which trigger an OB start.

③ An interrupt OB can only be interrupted by a time error interrupt OB (OB 80).

④ An new alarm-triggering event occurs during interrupt processing. This new event is added to a queue. The queued events successively call their corresponding OBs only after execution of the current interrupt OBs was completed and according to the following rules:

- Events are processed in the order of their priority (starting at the highest priority)
- Events of the same priority are processed in chronological order

⑤ The cyclic OBs are processed one after the other.

Notes on queues

- Every priority class (OBs of the same priority to be called) is assigned a separate queue. The size of those queues is set by default.
- Any new event leading to the overflow of a queue is discarded and therefore lost. A "time error interrupt event" is generated simultaneously. Information identifying the OB that caused the error is included in the start information of the time error interrupt OB (OB 80). A corresponding reaction such as an alarm trigger can be programmed in the time error interrupt OB.

Example of a hardware interrupt event

The function principle of event-oriented program execution in the S7-1200-CPU is described based on the example of a hardware interrupt-triggering module.

Process events and their priority

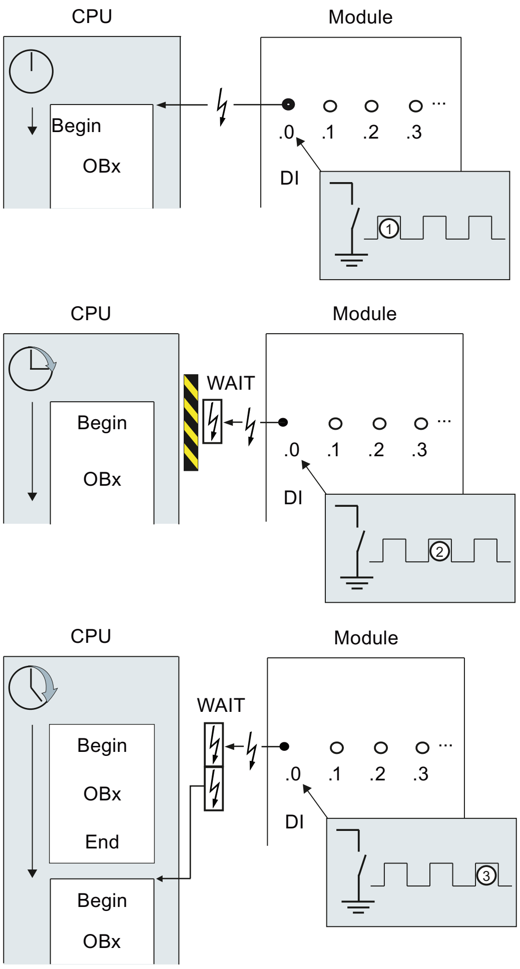
Process events are triggered by the I/O (e.g. at a digital input) and initiate a call of the assigned OB in the S7-1200 CPU. OBs assigned to a hardware interrupt event are so-called hardware interrupt OBs.

Examples of process events and their priority:

- Process events "rising edge" or "falling edge" at an interrupt-triggering module: The hardware interrupt OB started by such an event is always assigned priority 5.
- Process events from a high-speed counter
  - Count value corresponds to the reference value
  - Change count direction
  - External reset of the high-speed counter

    The hardware interrupt OB started by this event is always assigned priority 6.

The following figure shows the sequence of hardware interrupt execution.



Hardware interrupt execution

① A hardware interrupt-triggering event such as a rising edge at the input calls the OB to which it is assigned.

② If a new event occurs that triggers a hardware interrupt while the OB is executing, this event is entered in a queue.

③ The new event that triggers a hardware interrupt starts the hardware interrupt OB assigned to the event.

Assigning the interrupt-triggering event

The interrupt-triggering event is assigned to an OB in the input properties of the device view.

- An interrupt-triggering event can only be assigned to a single OB.
- OBs, however, can be assigned to several interrupt-triggering events.
  That is, you could assign both the rising and the falling edge event to the same interrupt OB in order to trigger the same reaction to any transition of the input signal.
- The started OB can interrupt a cycle OB at every instruction. Consistent data access is secured up to dword size.
- You can parameterize module-specific interrupt-triggering events such as a rising and the falling edge at the input.
- Assign the interrupt-triggering event and the OB to be started in the configuration of the interrupt-triggering module. However, within the started hardware interrupt OB you can override this assignment using the DETACH instruction, or assign the same event to a different OB using the ATTACH instruction. This functionality allows a flexible reaction to external process signals.

Setting the operating behavior

Changing properties of the modules

Default settings

When they leave the factory, all hardware components with parameters have default settings suitable for standard applications. These default values allow the hardware components to be used immediately without making any additional settings.

You can, however, modify the behavior and the properties of the hardware components to suit the requirements and circumstances of your application. Hardware components with settable parameters include, for example, communications modules and several analog and digital modules.

Setting and loading parameters

When you have selected a hardware component in the device or network view, you can set the properties in the Inspector window. When you save a device configuration with its parameters, data is generated that needs to be loaded on the CPU. This data is transferred to the relevant modules during startup.

Properties of the CPUs

The properties of the CPUs have special significance for system behavior. For example for a CPU you can set:

- Interfaces
- Inputs and outputs
- High-speed counters
- Pulse generators
- Startup behavior
- Time-of-day
- Protection level
- Bit memory for system and clock
- Cycle time
- Communications load

The entry possibilities specify what is adjustable and in which value ranges. Fields that cannot be edited are disabled or are not shown in the properties window.

Requirement

You have already arranged the hardware components for which you want to change properties on a rack.

Procedure

To change the properties and parameters of the hardware components, follow these steps:

1. In the device or network view, select the hardware component or interface that you want to edit.
2. Edit the settings for the selected object:
   - For example in the device view you can edit addresses and names.
   - In the Inspector window additional setting possibilities are available.

You do not need to confirm your entries, the changed values will be applied immediately.

See also

Editing properties and parameters (Page 345)

Introduction to loading a configuration (Page 523)

CPU properties

Overview of the CPU properties

Overview

The following table provides you with an overview of the CPU properties:

Group	Properties	Description
General	Project information	General information to describe the inserted CPU. Apart from the slot number, you can change all the settings
	Catalog information	Read-only information from the hardware catalog for this CPU.
PROFINET interface	General	Name and comment for this PROFINET interface. The name is limited to 110 characters.
	Ethernet addresses	Select whether the PROFINET interface is networked. If subnets have already been created in project, they can be selected in the drop-down list. If not, you can create a new subnet with the "Add new subnet" button. Information on the IP address, subnet mask, and IP router usage in the subnet is available in the IP protocol. If an IP router is used, the information about the IP address of the IP router is necessary.
	Extended	Name of and comment on the port of the Ethernet interface
	Time-of-day synchronization	Settings for time-of-day synchronization in the NTP time format. The NTP (network time protocol) is a general mechanism for synchronizing system clocks in local and global area networks. In NTP mode, the interface of the CPU sends time queries (in client mode) at regular intervals to NTP servers on the subnet (LAN) and the addresses must be set in the parameters here. Based on the replies from the server, the most reliable and most accurate time is calculated and synchronized. The advantage of this mode is that it allows the time to be synchronized across subnets. The accuracy depends on the quality of the NTP server being used.
DI#/DO#	General	Name of and comment on the integrated digital inputs of the CPU.

Group	Properties	Description
	Digital inputs	Input delays can be set for digital inputs. The input delays can be set in groups (in each case for 4 inputs). The detection of a rising and a falling edge can be enabled for each digital input. A name and a hardware interrupt can be assigned to this event. Depending on the CPU, pulse catches can be activated at various inputs. When the pulse catch is activated, even pulse edges that are shorter than the cycle time of the program are detected.
	Digital outputs	The reaction to a mode change from RUN to STOP can be set for all digital outputs: The state can either be frozen (corresponds to retain last value) or you set a substitute value ("0" or "1")
	I/O/Diagnostic addresses	The address space of the input and output addresses is specified as is the process image. The hardware ID of the device is displayed.
AI#	General	Name of and comment on the integrated analog inputs of the CPU.
	Analog inputs	During noise reduction, the specified integration time suppresses interference frequencies at the specified frequency (in Hz). The channel address, measurement type, voltage range, smoothing, and overflow diagnostics must be specified in the "Channel #" group. The measurement type and voltage range are set permanently to voltage, 0 to 10 V. Smoothing analog values provides a stable analog signal for further processing. Smoothing analog values can be useful with slow measured value changes, for example in temperature measurement. The measured values are smoothed with digital filtering. Smoothing is achieved by the module forming mean values from a specified number of converted (digitalized) analog values. The selected level (slight, medium, strong) decides the number of analog signals used to create the mean value. If overflow diagnostics is enabled, a diagnostics event is generated if an overflow occurs.
	I/O/Diagnostic addresses	The address space of the input addresses is specified as is the process image. The hardware ID of the device is displayed.
High-speed counter (HSC)	High-speed counter (HSC)#	High-speed counters are typically used to drive counting mechanisms. For description and parameter assignment, see: Configuring high-speed counters (Page 559)

Group	Properties	Description
Pulse generators (PTO/PWM)	PTO#/PWM#	A pulse generator is activated and can be initialized with project information. For the parameter assignment of an activated pulse generator, specify the usage as PWM (Pulse Width Modulation) or as PTO (Pulse Train Output). Specify the output source, time base, pulse width format, cycle time, and initial pulse width for PWM. A pulse output is specified as the hardware output. The PWM output is controlled by the CTRL_PWM instruction, see CTRL_PWM. Specify the output source for PTO. A pulse output and a direction output are specified as the hardware outputs. A PTO is operated together with a high-speed counter in the "motion axis" count mode and controlled by the Motion Control technology object, see Auto-Hotspot. The hardware ID is displayed in the I/O-diagnostic addresses and, if the PWM function is selected, the address space of the output addresses and the process image can be selected.
Startup	Startup type	Setting the start up behavior after cycling power. See: Principles of the STARTUP mode (Page 527)
Time-of-day	Local time and daylight-saving time	Setting of the time zone in which the CPU is operated and setting of the daylight-saving/standard time changeover.
Protection	Protection and password for read/write access	Setting options for the level of protection (Page 572)
System and clock memory bits	System memory bits and clock memory bits	You use system memory bits for the following scans: • Is the current cycle the first since cycling power? • Have there been any diagnostics state changes compared with the previous cycle? • Scan for "1" (high) • Scan for "0" (low) Clock memory bits change their values at specified periodic intervals. See: Using clock memory (Page 571)
Cycle time	Maximum cycle time and minimum cycle time.	See: Cycle time and maximum cycle time (Page 552)
Communications load	Maximum allocation of the cycle for communication (as a percentage)	See: Cycle loading by communications (Page 553)
I/O addresses overview	-	Tabular representation of all addresses used by the CPU for integrated inputs/outputs as well as for the inserted modules. Addresses that are not used by any module are represented as gaps. The view can be filtered according to: • Input addresses • Output addresses • Address gaps

See also

Specifying input and output addresses (Page 431)

Configuring high-speed counters (Page 559)

Setting options for the level of protection (Page 572)

Cycle time and maximum cycle time (Page 552)

Cycle loading by communications (Page 553)

Assigning parameters to hardware interrupt OBs (Page 581)

Principles of the STARTUP mode (Page 527)

Access to the I/O addresses (Page 540)

Using clock memory (Page 571)

Addressing modules (Page 430)

Special features during startup (Page 530)

Cycle time and maximum cycle time

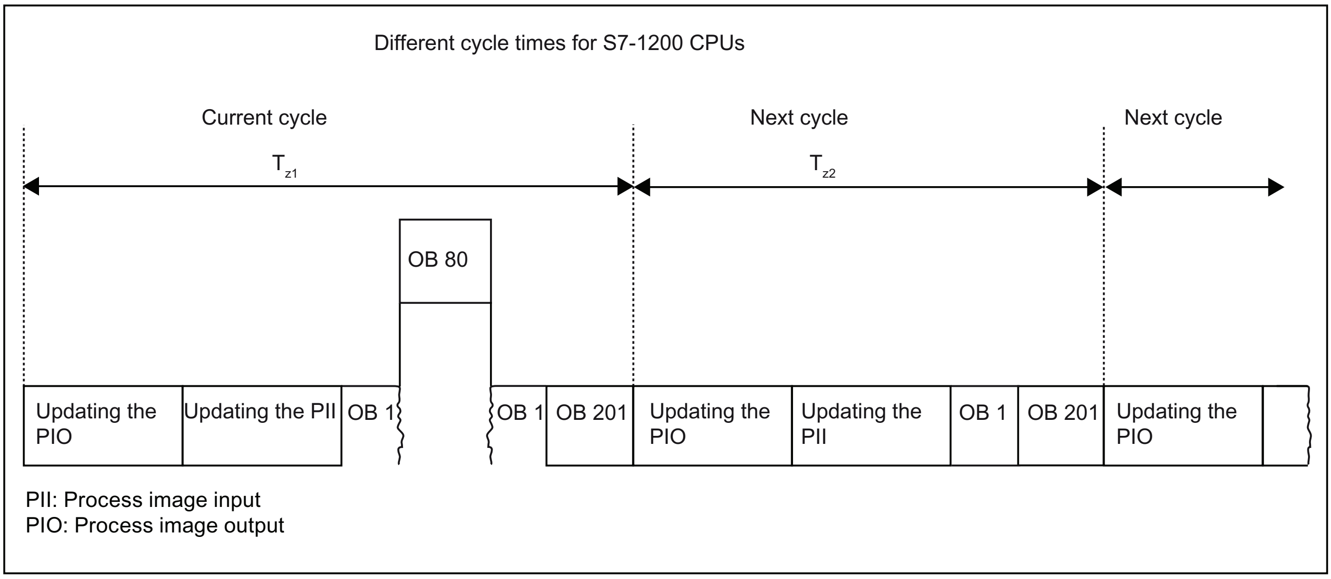
Function

The cycle time is the time that the operating system requires to execute the cyclic program and all the program sections that interrupt this cycle. The program execution can be interrupted by:

- Time errors and 2xMaxCycleTime errors
- System activities, e.g., process image updating

The cycle time (Tcyc) is therefore not the same for every cycle.

The following schematic shows an example of different cycle times (TZ1 ≠ TZ2) for S7-1200 CPUs:



In the current cycle, the cyclic OB used here (e.g. OB 1) will be interrupted by a time error (e.g. OB 80) Following the cyclic OB, the next cycle OB 201 is processed.

Maximum cycle time

The operating system monitors the execution time of the cyclic program for a configurable upper limit known as the maximum cycle time. You can restart this time monitoring at any point in your program by calling the RE_TRIGR instruction.

If the cyclic program exceeds the maximum cycle time, the operating system will attempt to start the time error OB (OB 80). If the OB is not available, the CPU ignores the overshoot of the maximum cycle time.

In addition to monitoring the runtime for overshooting of the maximum cycle time, adherence to a minimum cycle time is guaranteed. To do this, the operating system delays the start of the new cycle until the minimum cycle time has been reached. During this waiting time, new events and operating system services are processed.

If the maximum cycle time is exceeded a second time, for example while the time error OB is being processed (2xMaxCycleTime error), the CPU changes to STOP mode.

Cycle loading by communications

Function

The cycle time of the CPU can be extended due to communications processes. These communications processes include for example:

- Transferring data to another CPU
- Loading of blocks initiated by a programming device

You can control the duration of these communications processes to some extent using the CPU parameter "Cycle load due to communication".

In addition to communications processes, test functions also extend the cycle time. The "Cycle load due to communication" parameter can be used to influence the duration.

How the parameter works

You use the "Cycle load due to communication" parameter to enter the percentage of the overall CPU processing capacity that can be available for communications processes. This CPU processing capacity is now available at all times for communication. This processing capacity can be used for program execution when not required for communication.

Effect on the actual cycle time

The "Cycle load due to communication" parameter can be used to extend the cycle time of the cyclic organization block (e.g., OB 1) by a factor calculated according to the following formula:

$$\frac{100}{100 - \text{"Cycle load due to communication"}}$$

The formula does not take into account the effect of asynchronous events such as hardware interrupts or cyclic interrupts on the cycle time.

If the cycle time is extended due to communication processes, more asynchronous events may occur within the cycle time of the cyclic organization block. This extends the cycle still further. The extension depends on how many events occur and how long it takes to process them.

Example 1 – no additional asynchronous events:

If the "Cycle load due to communication" parameter is set to 50%, this can cause the cycle time of the cyclic organization block to increase by up to a factor of 2.

Example 2 – additional asynchronous events:

For a pure cycle time of 500 ms, a communication load of 50% can result in an actual cycle time of up to 1000 ms, provided that the CPU always has enough communications jobs to process. If, parallel to this, a cyclic interrupt with 20 ms processing time is executed every 100 ms, this cyclic interrupt would extend the cycle by a total of 5*20 ms = 100 ms without communication load. That is, the actual cycle time would be 600 ms. Because a cyclic interrupt also interrupts communications, it affects the cycle time by adding 10 * 20 ms at 50 % communication load. That is, in this case, the actual cycle time amounts to 1200 ms instead of 1000 ms.

Note

Observe the following:

- Check the effects of changing the value of the "Cycle load due to communication" parameter while the system is running.
- You must always consider the communication load when setting the minimum cycle time as time errors will otherwise occur.

Recommendations

- Increase this value only if the CPU is used primarily for communication purposes and the user program is not time critical.
- In all other situations you should only reduce this value.

Time-of-day functions

Basic principles of time of day functions

All S7-1200 CPUs are equipped with an internal clock. The backup supports the display of the correct time for up to 10 hours if the power supply is interrupted.

Time-of-day format

The clock always shows the time of day with a resolution of 1 millisecond and the date including the day of the week. The time adjustment for daylight-saving time is also taken into account.

Setting and reading the time of day

Setting and reading the time with instructions

You can set, start and read the time of day and date on the CPU clock with the following instructions in the user program:

- Set the time of day: "WR_SYS_T"
- Read the time of day "WR_SYS_T"
- Read local time "RD_LOC_T"
- Write local time "WRLT_DTL"

Manual setting

You can also read and set the time of day manually in the online and diagnostics view under "Functions > Set time of day".

Parameterizing the clock

Clock parameters

The clock parameters allow you to make the following settings:

- Enable time-of-day synchronization using an NTP server

  Select this check box if you want the internal clock to be synchronized using the NTP synchronization mode.
- Network time server

  The IP addresses of up to four NTP servers need to be configured.
- Update interval

  The update interval defines the interval between time queries.

High-speed counters

General information on high-speed counters

Introduction

High-speed counters are typically used to drive counting mechanisms in which a shaft turning at a constant speed is equipped with an incremental step encoder. The incremental step encoder ensures a certain number of count values per rotation and a reset pulse once per rotation. The clock memory bit(s) and the reset pulse of the incremental step encoder supply the inputs for the high-speed counter.

The various S7-1200 CPUs have differing numbers of high-speed counters available:

S7-1200 CPU	Number of HSCs	HSC designation
CPU 1211C	3 (with digital signal board 4)*	HSC1…3 (and HSC5)*
CPU 1212C	4 (with digital signal board 5)*	HSC1…4 (and HSC5)*
CPU 1214C	6	HSC1…6

* with DI2/DO2 signal board

How it works

The first of several default values is loaded on the high-speed counter. The required outputs are enabled for the time during which the current value of the counter is lower than the default value. The counter is set up so that an interrupt occurs if the current value of the counter is equal to the default value or when the counter is reset.

If the current value is equal to the default value and an interrupt event results, a new default value is loaded and the next signal state is set for the outputs. If an interrupt event occurs because the counter is reset, the first default value and the first signal states of the outputs are set and the cycle repeated.

Since the interrupts occur much less frequently than the high-speed counter counts, a precise control of the fast operations can be implemented with only a slight influence on the overall cycle of the automation system. Since you can assign specific interrupt programs to interrupts, each new default can be loaded in a separate interrupt program allowing simple control of the state.

Note

You can also process all interrupt events in a single interrupt program.

Count algorithms of the various counters

All counters work in the same way, however some high-speed counters do not support all count algorithms. There are four basic count algorithms:

- Single-phase counter with internal direction control
- Single-phase counter with external direction control
- 2-phase counter with 2 clock inputs
- A/B counter

Each high-speed counter can be used with or without a reset input. If the reset input is activated, this resets the current value. The current value remains reset until the reset input is deactivated.

See also

Configuring high-speed counters (Page 559)

Interdependency of the counter mode and counter inputs (Page 557)

Interdependency of the counter mode and counter inputs

General information on counter mode and counter inputs

You can assign not only the counter modes and counter inputs to the high-speed counters but also functions such as clock pulse generator, direction control, and reset. The following rules apply:

- An input cannot be used for two different functions.
- If an input is not required by the current counter mode of the defined high-speed counter, it can be used other purposes.

If, for example, you set HSC1 to counter mode 1, in which inputs I0.0 and I0.3 are required, you can use I0.1 for edge interrupts or for HSC2.

If, for example, you set HSC1 and HSC5, inputs I0.0 (HSC1) and I1.0 (HSC5) are always used with the counting and frequency counter modes. As a result, these two inputs are not available for any other functions when counters are operated.

You have additional inputs available if you use a digital signal board.

Overview of the interdependency of counter mode and counter inputs

Counter mode	Description	Inputs		
	HSC1	I0.0 (CPU) I4.0 (signal board)	I0.1 (CPU) I4.1 (signal board)	I0.3 (CPU) I4.3 (signal board)
	HSC2	I0.2 (CPU) I4.2 (signal board)	I0.3 (CPU) I4.3 (signal board)	I0.1 (CPU) I4.1 (signal board)
	HSC3*	I0.4 (CPU)	I0.5 (CPU)	I0.7 (CPU)
	HSC4 (CPU 1212/14C only)	I0.6 (CPU)	I0.7 (CPU)	I0.5 (CPU)
	HSC5 (CPU 1214C only)**	I1.0 (CPU) I4.0 (signal board)	I1.1 (CPU) I4.1 (signal board)	I1.2 (CPU) I4.3 (signal board)
	HSC6 (CPU 1214C only)**	I1.3 (CPU)	I1.4 (CPU)	I1.5 (CPU)
Counting / frequency	Single-phase counter with internal direction control	Clock pulse generator		
Counting		Clock pulse generator		Resetting
Counting / frequency	Single-phase counter with external direction control	Clock pulse generator	Direction	
Counting		Clock pulse generator	Direction	Resetting
Counting / frequency	2-phase counter with 2 clock inputs	Clock pulse generator forwards	Clock pulse generator backwards	
Counting		Clock pulse generator forwards	Clock pulse generator backwards	Resetting
Counting / frequency	A/B counter	Clock pulse generator A	Clock pulse generator B	
Counting		Clock pulse generator A	Clock pulse generator B	Resetting
Motion axis	Pulse generators PWM/PTO	HSC1 and HSC2 support the motion axis count mode for the PTO1 and PTO2 pulse generators: • For PTO1, HSC1 evaluates the Q0.0 outputs for the number of pulses. • For PTO2, HSC2 evaluates the Q0.2 outputs for the number of pulses. Q0.1 is used as the output for the motion direction.		

* HSC3 can only be used for CPU 1211 without a reset input

** HSC5 can be also be used for CPU 1211/12 if a DI2/DO2 signal board is used

See also

General information on high-speed counters (Page 556)

Configuring high-speed counters (Page 559)

Configuring high-speed counters

Requirement

An S7-1200 CPU has been inserted in the hardware configuration.

Procedure

To configure a high-speed counter, follow these steps:

1. Select an S7-1200 CPU in the device or network view.
2. Click on the required high-speed counter under "Properties > High-speed counter" in the Inspector window:
   - CPU 1211C: HSC1 to HSC3 (also HSC5 with a DI2/DO2 signal board)
   - CPU 1212C: HSC1 to HSC4 (also HSC5 with a DI2/DO2 signal board)
   - CPU 1214C: HSC1 to HSC6
3. Enable the high-speed counter in the "General" parameter group using the relevant check box.

   Note

   If you use a CPU 1211C or CPU 1212C with a DI2/DO2 signal board, you can also enable the high-speed counter HSC5.

   Note

   If you activate the pulse generators and operate them as PTO1 or PTO2, they use the associated high-speed counter HSC1 or HSC2 with "Motion axis" counting mode to evaluate the hardware outputs. If you configure high-speed counter HSC1 or HSC2 for other counting tasks, these cannot be used by pulse generator PTO1 or PTO2, respectively.

   If required, you can enter a name and a comment for the high-speed counter here.
4. Define the functionality of the high-speed counter in the "Function" parameter group:
   - Count mode: Select what you want to be counted from the drop-down list.
   - Operating phase: Select the count algorithm from the drop-down list.
   - Input source: Select the on-board CPU inputs or the inputs of an optional digital signal board as the input source for the count pulses from the drop-down list.
   - Count direction is specified by: If you have selected a single-phase operating phase, open the drop-down list and select whether the count direction is set internally by an SFB parameter of the user program or externally via a digital input.
   - Initial count direction: If the user program is set as the internal direction control for the count direction, you can select the count direction at the start of counting from the drop-down list.
   - Frequency meter period: If frequency is set as the count mode, you can select the duration of the frequency meter periods in the drop-down list.

5. Specify the initial values and reset condition of the high-speed counter in the "Reset to initial values" parameter group:
   - Initial counter value: Enter a start value for the high-speed counter.
   - Initial reference value: Enter a maximum value for the high-speed counter.

   Here, you can also specify whether the high-speed counter will use a reset input and the set the corresponding signal level for the reset input from the drop-down list.
6. Configure the reaction of the high-speed counter to certain events in the "Event configuration" parameter group. The following events can trigger an interrupt:
   - The counter value matches the reference value.
   - An external reset event was generated.
   - A change of direction was triggered.

   Enable an interrupt reaction using the check box, enter a name and select a hardware interrupt for the interrupt from the drop-down list.
7. Assign the start address for the high-speed counter in the "I/O/Diagnostic addresses" parameter group.

   Note

   In the "Hardware inputs" parameter group, you can only see which hardware inputs and values are being used for the clock, direction determination, reset pulse, and maximum count speed.

Result

You have now adapted the parameters of the high-speed counter to the requirements of your project.

See also

General information on high-speed counters (Page 556)

Interdependency of the counter mode and counter inputs (Page 557)

Point-to-point communication

Overview of point-to-point communication

PtP communication is communication via a serial interface that uses standardized UART data transmission (Universal Asynchronous Receiver/Transmitter). The S7-1200 uses communications modules with an RS-232 or RS-485 interface to establish PtP communication.

Functions of point-to-point communication

Point-to-point communication (PtP) allows numerous applications:

- Direct transmission of information to an external device, for example a printer or a barcode reader
- Reception of information from external devices such as barcode readers, RFID readers, cameras and third-party optical systems as well as many other devices.
- Exchange of information with third-party devices, for example GPS devices, radio modems and many others

The Freeport protocol

The S7-1200 supports the Freeport protocol for character-based serial communication. Using Freeport communication, the data transmission protocol can be configured entirely by the user program.

Siemens provides libraries with Freeport communication functions that you can use in your user program:

- USS Drive protocol
- Modbus RTU Master protocol
- Modbus RTU Slave protocol

See also

Configuring a communications port (Page 562)

Using RS-232 and RS-485 communications modules

Communications modules with RS-232 and RS-485 interfaces

In an S7-1200 CPU, you can use two different communications modules:

- RS-232 communications module
- RS-485 communications module

The communications modules can be connected to the S7-1200 CPU via the I/O channel on the left. You can plug in up to three different modules.

Properties of the communications modules

The communications modules have the following features:

- Support of the Freeport protocol
- Configuration by the user program with the aid of expanded instructions and library functions

Configuring a communications port

Configuring a communications port

After you have inserted a communications module with an RS-232 or RS-485 interface, you then set the interface parameters. You set the parameters for the interface either in the properties of the interface or you control the interface parameters from the user program using the PORT_CFG instruction. The following description relates to the graphic configuration.

Note

If you use the user program to change the port setting, the settings of the graphic configuration are overwritten.

You should also keep in mind that the settings made by the user program are not retained if there is a power down.

Requirement

- A communications module is already plugged in.
- You are in the device view.

Procedure

To configure the communications port, proceed as follows:

1. Select the interface in the graphic representation in the device view.

   The properties of the interface are displayed in the Inspector window.
2. Select the "Port configuration" group in the area navigation of the Inspector window.

   The settings of the port are displayed.
3. From the "Transmission speed" drop-down list, select the speed for the data transmission. With user-programmed communication, remember the influence of the transmission speed on the changeover time.
4. From the "Parity" drop-down list, select the type of detection of bad information words.
5. Using the "Data bits" drop-down list, decide whether a character consists of eight or seven bits.
6. From the "Stop bit" drop-down list, select how many bits will identify the end of a transmitted word.

7. From the "Flow control" drop-down list, select the method for ensuring a trouble-free data stream between sender and receiver. This parameter can only be set for the RS-232 interface.
   – Enter a HEX value in the "XON character" box that will cause the transmission of the message to be continued when it is detected. This parameter can only be set for software-controlled data flow control.
   – Enter a HEX value in the "XOFF character" box that will cause the transmission of the message to be suspended for the set wait time. This parameter can only be set for software-controlled data flow control.
8. In the "Wait time" box, enter a wait time in ms that must be kept to after the end of the message before the next transmission can start.

Note

You can configure the interface in the network view as well. To do so, you must first select the communication module in the tabular network view and then select the interface in the Inspector window. Then you can continue as described above.

See also

Setting data flow control (Page 563)

Setting data flow control

Data flow control

Data flow control is a method that ensures a balanced send and receive behavior. In the ideal situation, the intelligent control does not allow data to be lost. It ensures that a device does not send more information than the receiving partner can process.

There are two methods of data flow control:

- Hardware-controlled data flow control
- Software-controlled data flow control

With both methods, the DSR signals of the communications partners must be active at the beginning of transmission. If the DSR signals are inactive, the transmission is not started.

The RS-232 communications module can handle both methods. The RS-485 communications module does not support data flow control.

Hardware-controlled data flow control

Hardware-controlled data flow control uses the request to send (RTS) and clear to send (CTS) signals. With the RS-232 communications module, the RTS signal is transmitted via the output by pin 7. The CTS signal is received via pin 8.

If hardware-controlled data flow control is enabled, the RTS signal is then always set to activated when data is sent. At the same time, the CTS signal is monitored to check whether the receiving device can accept data. If the CTS signal is active, the module can transfer data until the CTS signal becomes inactive. If the CTS signal is inactive, the data transfer must be suspended for the set wait time. If the CTS signal is still inactive after the set wait time, the data transfer is aborted and an error is signaled back to the user program.

Data flow control using hardware handshaking

If data flow control is controlled by hardware handshaking, the sending device sets the RTS signal to active as default. A device such as a modem can then transfer data at any time. It does not wait for the CTS signal of the receiver. The sending device itself monitors it own transmission by sending only a limited number of frames (characters), for example to prevent overflow of the receive buffer. If there is nevertheless an overflow, the transferring device must hold back the message and signal an error back to the user program.

Software-controlled data flow control

Software-controlled data flow control uses certain characters within the messages and these control the transfer. These characters are ASCII characters selected for XON and XOFF.

XOFF indicates when a transmission must be suspended. XON indicates when a transmission can be continued.

If the sending device receives the XOFF character, it must suspend sending for the selected wait time. If the XON character is sent after the selected wait time, the transfer is continued. If no XON character is received after the wait time, an error is signaled back to the user program.

Software data flow control requires full duplex communication because the receiving partner needs to send the XON character during the ongoing transfer.

See also

Configuring a communications port (Page 562)

Configuration of message transfer

User-programmed communication

You can control the data traffic between a communications module and a device connected externally via the serial interface using your own mechanisms. If you want to do this, you will need to define a communications protocol yourself. In freely programmable communication, ASCII and binary protocols are supported for message transfer.

Within the communications protocol, you will need to specify the criteria by which the start and end of a transferred message can be recognized in the data stream.

User-programmed communication can only be activated in RUN mode. If there is a change to STOP mode, the user-programmed communication is stopped.

Specifying the communications protocol

You can specify the communications protocol as follows:

- With the user program
  - The behavior when sending data is controlled by the SEND_CFG instruction.
  - The behavior when receiving data is controlled by the RCV_CFG instruction.
- Using parameter settings set graphically in the Inspector window

Note

If you change the communications protocol from the user program, the settings of the graphic configuration are overwritten.

You should keep in mind that the settings made by the user program are not retained if there is a power down.

See also

User-programmed communication with RS-232 devices (Page 566)

Making the settings for sending (Page 568)

Specifying the start of the message (Page 568)

Specifying the end of the message (Page 569)

User-programmed communication with RS-232 devices

RS-232/PIP multi-master cable and user-programmed communication with RS-232 devices

Using the RS-232/PIP multi-master cable and user-programmed communication, you can connect a wide variety of RS-232-compliant devices to the communications modules of the S7-1200. The cable must, however, be set to the "PIP/user-programmed communication" mode.

Settings on the cable

The switches on the cable must be set as follows:

- Switch 5 must be set to 0
- Switch 6 sets either the local mode (DCE) or the remote mode (DTE):
  - Switch set to 0 for the local mode
  - Switch set to 1 for the remote mode

Changing over between send and receive mode

The RS-232/PIP multi-master cable is in send mode when data is sent from the RS-232 interface to the RS-485 interface. The cable is in receive mode when it is idle or when data is sent from the RS-485 interface to the RS-232 interface. The cable changes from receive to send mode immediately when it detects characters on the RS-232 send line.

Supported transmission speeds

The RS-232/PIP multi-master cable supports transmission rates between 1200 baud and 115.2 kbaud. The RS-232/PIP multi-master cable can be set to the required transmission rate using the DIP switch on the PC/PIP cable.

The following table shows the switch settings for the various transmission speeds:

Transmission speed	Switchover time	Settings (1 = up)
115200 bps	0.15 ms	110
57600 bps	0.3 ms	111
38400 bps	0.5 ms	000
19200 bps	1.0 ms	001
9600 bps	2.0 ms	010
4800 bps	4.0 ms	011
2400 bps	7.0 ms	100
1200 bps	14.0 ms	101

The cable returns to receive mode when the RS-232 send line is idle for a certain time that is defined as the changeover time of the cable. The set transmission speed influences the changeover time as shown in the table.

Influence of the changeover time

When working with an RS-232/PIP multi-master cable in a system in which user-programmed communication is used, the program must take into account the changeover time for the following reasons:

- The communications module reacts to messages sent by the RS-232 device.

  Once the communications module has received a request from the RS-232 device, it must delay the reaction message for a period that is equal to or longer than the changeover time of the cable.

- The RS-232 device reacts to messages sent by the communications module.

  Once the communications module has received a reaction message from the RS-232 device, it must delay the next request message for a period that is equal to or longer than the changeover time of the cable.

In both situations, the RS-232-PIP multi-master cable has enough time to change from send to receive mode so that the data can be sent from the RS-485 interface to the RS-232 interface.

See also

Configuration of message transfer (Page 565)

Making the settings for sending (Page 568)

Specifying the start of the message (Page 568)

Specifying the end of the message (Page 569)

Making the settings for sending

Sending messages

You can program pauses between individual messages.

The following table shows which pauses can be set:

Parameter	Definition
RTS ON delay	You can set the time that must elapse after the send request RTS (request to send) before the actual data transfer starts.
RTS OFF delay	You can set the time that must elapse after the complete transfer before RTS signal is deactivated.
Send pause at the start of the message	You can specify that a pause is sent at the start of every message transfer when the RTS ON delay has elapsed. The pause is specified in bit times.
Send Idle Line after a pause	You can make a setting so that following a selected pause at the start of the message, the "Idle Line" signal is output to signal that the line is not in use. To enable the parameter, "Send pause at message start" must be set. The duration of the "Idle Line" signal is specified in bit times.

See also

Specifying the start of the message (Page 568)

Specifying the end of the message (Page 569)

User-programmed communication with RS-232 devices (Page 566)

Specifying the start of the message

Recognizing the start of the message

To signal to the receiver when the transfer of a message is completed and when the next message transfer starts, criteria must be specified in the transmission protocol to identify the end and start of a message.

If a criterion is met that indicates the start of a message, the receiver starts searching the data stream for criteria that mean the end of the message.

There are two different methods for identifying the start of a message:

- Starting with any character:

  Any character can defined the start of a message. This is the default method.

- Starting with a specific condition:

  The start of a message is identified based on selected conditions.

Conditions for detecting the start of a message

The following table shows the various options for defining the start of a message:

Parameter	Definition
Recognize start of message by line break	The receiver recognizes a line break when the received data stream is interrupted for longer than one character. If this is the case, the start of the message is identified by the line break.
Recognize start of message by idle line	The start of a message is recognized when the send transmission line is in the idle state for a certain time (specified in bit times) followed by an event such as reception of a character.
Recognize start of message with individual characters	The start of a message is recognized when a certain character occurs. You can enter the character as a HEX value.
Recognize start of message by a character string	The start of a message is detected when one of the specified character sequences arrives in the data stream. You can specify up to four character sequences each with up to five characters.

The individual conditions can be logically linked in any way.

See also

Making the settings for sending (Page 568)

User-programmed communication with RS-232 devices (Page 566)

Specifying the end of the message

Recognizing the end of the message

To signal to the receiver when the transfer of a message is completed and when the next message transfer starts, criteria must be specified in the transmission protocol to identify the end and start of a message.

In total, there are six different methods of recognizing the end of a message and these can all be logically linked in any way. The following table shows the various possible setting options:

Parameter	Definition
Recognize end of message by message timeout	The end of a message is recognized automatically when a selected maximum duration for a message is exceeded. Values from 0 to 65535 ms can be set.
Recognize end of message by reply timeout	The end of a message is recognized when there is no reply within a set time after transferring data. Values from 0 to 65535 ms can be set.

Parameter	Definition
Recognize end of message by timeout between characters	The end of a message is detected when the time between two characters specified in bit times is exceeded. Values from 0 to 2500 bit times can be set. The S7-1200 CPU only accepts a maximum time of eight seconds even if the value that is set results in a duration of more than eight seconds.
Recognize end of message by maximum length	The end of a message is recognized when the maximum length of a message is exceeded. Values from 1 to 1023 characters can be set.
Read message length from message	The message itself contains information about the length of the message. The end of a message is reached when the value taken from the message is reached. Which characters are used for the evaluation of the message length is specified with the following parameters: • Offset of the length field in the message The value decides the position of the character in the message that will be used to indicate the message length. Values from 1 to 1022 characters can be set. • Size of the length field This value specifies how many characters starting at the first evaluation position will be used to indicate the message length. Values of 0, 1, 2 and 4 characters can be set. • The data following the length field (does not belong to the message length) The value specifies the number of bytes after the length field that must be ignored in the evaluation of the message length. Values from 0 to 255 characters can be set.
Recognize message length by a character string	The end of a message is detected when the specified character sequence arrives in the data stream. You can define up to five characters in the character string.

See also

Making the settings for sending (Page 568)

User-programmed communication with RS-232 devices (Page 566)

Using clock memory

Clock memory

A clock memory is a bit memory that changes its binary status periodically in the pulse-no-pulse ratio of 1:1.

You decide which memory byte of the CPU will become the clock memory byte when assigning the clock memory parameters.

Benefits

You can use clock memory, for example, to activate flashing indicator lamps or to initiate periodically recurring operations such as recording of actual values.

Available frequencies

Each bit of the clock bit memory byte is assigned a frequency. The following table shows the assignment:

Bit of the clock memory byte	7	6	5	4	3	2	1	0
Period (s)	2.0	1.6	1.0	0.8	0.5	0.4	0.2	0.1
Frequency (Hz)	0.5	0.625	1	1.25	2	2.5	5	10

Note

Clock memory runs asynchronously to the CPU cycle, i.e. the status of the clock memory can change several times during a long cycle.

The selected memory byte cannot be used for intermediate storage of data.

Enabling system memory

System memory

A system memory is a bit memory with defined values.

You decide which memory byte of the CPU will become the system memory byte when assigning the system memory parameters.

Benefits

You can use system memory in the user program, for example to run program segments in only the first program cycle after start-up. Two system memory bits are constant 1 or constant 0.

Bits of the system memory bytes

The following table shows the meaning of the system memory:

Bit of the system memory bytes	7	6	5	4	3	2	1	0
Meaning	Reserved (=0)	Reserved (=0)	Reserved (=0)	Reserved (=0)	=0	=1	=1 with change to the diagnosis status	=1 in first program cycle after startup, otherwise 0

Note

The selected memory byte cannot be used for intermediate storage of data.

Setting options for the level of protection

Protection level

In the following, you will learn how to use the various protection levels of the S7-1200 CPU.

Effects of the protection level setting

You can choose between the following protection levels:

- No protection: This corresponds to the default behavior. You cannot enter a password. Read and write access is always permitted.
- Write protection: Only read-only access is possible. You cannot change any data on the CPU and cannot load any blocks or a configuration. PLC tags that you have marked as relevant for HMI are excluded from the write protection. Assignment of a password is required to select this protection level.
- Write/read protection No write or read access is possible in the "Accessible devices" area or in the project for devices that are switched online. Only the CPU type and the identification data can be displayed in the project tree under "Accessible devices". Display of online information or blocks under "Accessible devices", or in the project for devices interconnected online, is possible.

  The following objects are excluded from write/read protection:

  – PLC tags that are marked as HMI-relevant

  – Monitorable properties of HMI objects

  Assignment of a password is required to select this protection level.

Behavior of a password-protected CPU during operation

The CPU protection takes effect after the settings are downloaded to the CPU

Before an online function is executed, the necessary permission is checked and, if necessary, the user is prompted to enter a password.

Example: The module was assigned a protection level and you want to execute the "Modify tags" function. This requires write access; therefore, the assigned password must be entered to execute the function.

The functions protected by a password can only be executed by one programming device/PC at any one time. Another programming device/PC cannot log on with a password.

Access authorization to the protected data is in effect for the duration of the online connection, or until the access authorization is manually rescinded with "Online > Delete access rights". Access authorization will also expire when the project is closed.

Note

You can not restrict functions for process control, monitoring, and communications.

Some functions are still protected due to their use as online data. RUN/STOP in the "Online Tools" task card or "Set time of day" in the diagnostics and online editor is therefore write-protected.

Organization blocks

Organization blocks for startup

Description

You can determine the boundary conditions for the startup behavior of your CPU, for example the initialization values for "RUN". To do this, write a startup program. The startup program consists of one or more startup OBs (OB numbers 100 or >= 123).

The startup program is executed once during the transition from "STOP" mode to "RUN" mode. Current values from the process image of the inputs are not available for startup program, nor can these values be set.

After the complete execution of the startup OBs, the process image of the inputs is read in and the cyclic program is started.

There is no time limit for executing the startup program. Therefore the scan cycle monitoring time is not active. Time-driven or interrupt-driven organization blocks cannot be used.

Start information

A startup OB has the following start information:

Tag	Data type	Description
LostRetentive	BOOL	= 1, if retentive data storage areas have been lost
LostRTC	BOOL	= 1, if realtime clock has been lost

See also

Events and OBs (Page 540)

Organization blocks for cyclic program execution

Introduction

For the program execution to start, at least one program cycle OB must be present in the project. The operating system calls this program cycle OB once in each cycle and thereby starts the execution of the user program. You can use multiple OBs (OB numbers >= 123).

The program cycle OBs have the priority class 1. This corresponds to the lowest priority of all OBs. The cyclic program can be interrupted by events of any other event class.

Programming cyclic program execution

You program cyclic program execution by writing your user program in the cycle OBs and the blocks that they call.

The first cyclic program execution begins as soon as the startup program has ended without errors. The cycle restarts after the end of each cyclic program execution.

Sequence of cyclic program execution

One cycle of the program execution encompasses the following steps:

1. The operating system starts the scan cycle monitoring time.
2. The operating system writes the values from the process image output to the output modules.
3. The operating system reads out the state of the inputs of the input modules and updates the process image input.
4. The operating system processes the user program and executes the operations contained in the program.
5. At the end of a cycle, the operating system executes any tasks that are pending, for example, loading and deleting blocks, or calling other cycle OBs.
6. Finally, the CPU returns to the start of the cycle and restarts the scan cycle monitoring time.

See also: process image input/output (Page 537)

Options for interrupting

Cyclic program execution can be interrupted by the following events:

- An interrupt
- A STOP command, triggered by
  - Operation of the programming device
  - "STP" instruction
- Supply voltage failure
- Occurrence of a device fault or program error

Start information

Cycle OBs have no start information.

See also

Events and OBs (Page 540)

Organization blocks for interrupt-driven program execution

Organization blocks for time-delay interrupts

Description

A time-delay interrupt OB is started after a configurable time delay of the operating system. The delay time starts after the SRT_DINT instruction is called.

You can use up to four time-delay interrupt OBs or cyclic OBs (OB numbers >= 123) in your program. If, for example, you are already using two cyclic interrupt OBs, you can insert a maximum of two further time-delay interrupt OBs in your program.

You can use the CAN_DINT instruction to prevent the execution of a time-delay interrupt that has not yet started.

Function of time-delay interrupt OBs

The operating system starts the corresponding OB after the delay time, which you have transferred with an OB number and an identifier to the SRT_DINT instruction.

To use a time-delay interrupt OB, you must execute the following tasks:

- You must call the instruction SRT_DINT.
- You must download the time-delay interrupt OB to the CPU as part of your program.

The delay time is measured with a precision of 1 ms. A delay time can immediately start again after it expires.

Time delay interrupt OBs are executed only when the CPU is in the "RUN" mode. A warm restart clears all start events of time-delay interrupt OBs.

The operating system calls the time-delay interrupt OB if one of the following events occurs:

- If the operating system attempts to start an OB that is not loaded and you specified its number when calling the SRT_DINT instruction.
- If the next start event for a time-delay interrupt occurs before the time delay OB has completely executed.

You can disable and re-enable time-delay interrupts using the DIS_AIRT and EN_AIRT instructions.

Note

If you disable an interrupt with DIS_AIRT after executing SRT_DINT, this interrupt executes only after it has been enabled with EN_AIRT. The delay time is extended accordingly.

Start information

Time delay interrupt OBs have no start information.

See also

Events and OBs (Page 540)

Organization blocks for cyclic interrupts

Description

Cyclic interrupt OBs serve to start program in periodic time intervals independently of the cyclic program execution. The start times of a cyclic interrupt OB are specified using the time base and the phase offset.

The time base defines the intervals at which the cyclic interrupt OB is started and is a whole multiple of the basic clock cycle of 1 ms. The phase offset is the time by which the start time is offset compared with the basic clock cycle. If several cyclic interrupt OBs are being used, you can use this offset to prevent a simultaneous start time if the time bases of the cyclic interrupt OBs have common multiples.

You can specify a time period between 1 ms and 60000 ms.

You can use up to four cyclic interrupt OBs or time-delay OBs (OB numbers >= 123) in your program. If, for example, you are already using two time-delay interrupt OBs, you can insert a maximum of two further cyclic interrupt OBs in your program.

Note

The execution time of each cyclic interrupt OB must be noticeably smaller than its time base. If a cyclic interrupt OB has not been completely executed but execution is again pending because the cycle clock has expired, the time error interrupt OB is started. The cyclic interrupt that caused the error is executed later or discarded.

Example of the use of phase offset

You have inserted two cyclic interrupt OBs in your program:

- Cyclic interrupt OB1
- Cyclic interrupt OB2

For cyclic interrupt OB1, you have set a time base of 20 ms and for cyclic interrupt OB2 a time base of 100 ms. After the expiry of the time base of 100 ms, the cyclic interrupt OB1 reaches the start time for the fifth time, cyclic interrupt OB2 for the first time. To nevertheless execute the cyclic interrupt OBs offset, enter a phase offset for one of the two cyclic interrupt OBs.

Start information

Cyclic interrupt OBs have no start information.

See also

Assigning parameters to cyclic interrupt OBs (Page 583)

Events and OBs (Page 540)

Organization blocks for hardware interrupts

Description

You can use hardware interrupt OBs to react to specific events. You can assign an event that triggers an alarm to precisely one hardware interrupt OB. A hardware interrupt OB in contrast can be assigned to several events.

Hardware interrupts can be triggered by high-speed counters and input channels. For each high-speed counter and input channel that should trigger a hardware interrupt, the following properties need to be configured:

- The process event that should trigger the hardware interrupt (for example, the change of a count direction of a high-speed counter)
- The number of the hardware interrupt OB which is assigned to this process event

You can use up to 50 hardware interrupt OBs (OB numbers >= 123) that are independent of each other in your program.

Function of a hardware interrupt OB

After triggering a hardware interrupt, the operating system identifies the channel of the input or the high-speed counter and determines the assigned hardware interrupt OB.

If no other interrupt OB is active, the determined hardware interrupt OB is called. If a different interrupt OB is already being executed, the hardware interrupt is placed in the queue of its priority class. The hardware interrupt is acknowledged after the completion of the assigned hardware interrupt OB.

If another event that triggers a hardware interrupt occurs on the same module during the time between identification and acknowledgement of a hardware interrupt, the following applies:

- If the event occurs on the channel that previously triggered the hardware interrupt, then no additional hardware interrupt is triggered. Another hardware interrupt can only be triggered if the current hardware interrupt is acknowledged.
- If the event occurs on a different channel, a hardware interrupt is triggered.

Hardware interrupt OBs are called only in the CPU's "RUN" mode.

Start information

Hardware interrupt OBs have no start information.

See also

Assigning parameters to hardware interrupt OBs (Page 581)

Events and OBs (Page 540)

Organization block for time error

Description

The operating system calls the time error interrupt OB (OB 80) if one of the following events occurs:

- The cyclic program exceeds the maximum cycle time.
- The OB called is currently still being executed (possible for time-delay interrupt OBs and cyclic interrupt OBs).
- An overflow has occurred in an interrupt OB queue.
- An interrupt was lost due to the excessive interrupt load.

If you have programmed no time error interrupt OB, the S7-1200 CPU reacts as follows:

- CPUs with firmware version V1.0: The CPU remains in RUN mode.
- CPUs with firmware version V2.0:
  - The CPUs goes to STOP mode when the maximum cycle time is exceeded.
  - With all other start events of the time error interrupt OB, the CPU remains in RUN mode.

With CPUs with firmware version V1.0, the two-time overshooting of the maximum cycle time does not lead to the calling of an OB, but to the STOP of the CPU. You can avoid the second violation by restarting the cycle monitoring of the CPU with the RE_TRIGR instruction.

You can use only one time error interrupt OB in your program.

Start information

The time error interrupt OB has the following start information:

Tag	Data type	Description
fault_id	BYTE	• 0x01: Maximum cycle time exceeded • 0x02: Called OB is still being executed • 0x07: Queue overflow • 0x09: Interrupt loss due to high interrupt load
csg_OBnr	OB_ANY	Number of the OB being executed at the time of the error
csg_prio	UINT	Priority of the OB being executed at the time of the error

See also

Events and OBs (Page 540)

Organization block for diagnostic error interrupts

Description

You can enable the diagnostic error interrupt for diagnostics-capable modules so that the module detects changes in the I/O status. As a result, the module triggers a diagnostic error interrupt in the following cases:

- A fault is present (incoming event)
- A fault is no longer present (outgoing event)

If no other interrupt OB is active, then the diagnostic error interrupt OB (OB 82) is called. If another interrupt OB is already being executed, the diagnostic error interrupt is placed in the queue of its priority group.

You can use only one diagnostic error interrupt OB in your program.

Start information

The diagnostic error interrupt OB has the following start information:

Tag	Data type	Description
IO_state	WORD	Contains the I/O status of the diagnostics-capable module.
laddr	HW_ANY	HW-ID
Channel	UINT	Channel number
multi_error	BOOL	= 1, if there is more than one error

IO_state tag

The following table shows the possible I/O states that the IO_state tag can contain:

IO_state	Description
Bit 0	Configuration correct: = 1, if the configuration is correct = 0, if the configuration is no longer correct
Bit 4	Error: = 1, if an error is present, e.g., a wire break = 0, if the error is no longer present
Bit 5	Configuration not correct: = 1, if the configuration is not correct = 0, if the configuration is correct again
Bit 6	I/O cannot be accessed: = 1, if an I/O access error has occurred In this case, laddr contains the hardware identifier of the I/O with the access error. = 0, if the I/O can be accessed again

See also

Events and OBs (Page 540)

Block parameters of organization blocks

Basics of block parameters

Introduction

Several organization blocks (OBs) have properties with which you can control their behavior or their assignment to specific events. You can influence these properties by assigning parameters.

Overview

You can assign parameters to the properties for the following organization blocks:

- Hardware interrupt OBs
- Cyclic interrupt OBs

See also

Assigning parameters to hardware interrupt OBs (Page 581)

Assigning parameters to cyclic interrupt OBs (Page 583)

Assigning parameters to hardware interrupt OBs

Introduction

You must select the corresponding event and assign the following parameters for every input channel and high-speed counter that should trigger a hardware interrupt:

- Event name
- Number of the hardware interrupt OB that is assigned to this process event

The parameters of the hardware interrupt are assigned in the properties of the corresponding device. You can assign parameters for up to 50 hardware interrupt OBs.

You can create the hardware interrupt OB to be assigned parameters either before or during activation of an event.

Procedure

To assign parameters for the hardware interrupt OB, follow these steps:

1. Double-click the "Devices & Networks" command in the project tree.

   The hardware and network editor opens in the network view.
2. Change to the device view.
3. If the Inspector window closed in the device view, select the "Inspector window" check box in the "View" menu.

   The Inspector window opens.
4. Click the "Properties" tab.
5. In the device view, select the module for which you want to a assign a hardware interrupt.
6. Select the corresponding event.
7. Enter an event name.
8. Select an existing hardware interrupt OB from the "Hardware interrupt" drop-down list.

   Note

   If you have not previously created any hardware interrupt OB, you can click "Add new block" in the drop-down list.

   See also: Creating organization blocks (Page 844)

   ---
9. If you want to assign further hardware interrupts, repeat steps 5 to 8.

See also

Basics of block parameters (Page 581)

Organization blocks for hardware interrupts (Page 578)

Events and OBs (Page 540)

Assigning parameters to cyclic interrupt OBs

Introduction

You can use cyclic interrupt OBs to start programs at regular time intervals. To do so you must enter a scan time and a phase shift for each cyclic interrupt OB used.

You can use up to four cyclic interrupt OBs or time-delay OBs (OB numbers >= 200) in your program. If, for example, you are already using two time-delay interrupt OBs, you can insert a maximum of two further cyclic interrupt OBs in your program.

Note

If you assign multiple cyclic OBs, make sure that you assign a different cycle time or phase offset to each cyclic interrupt OB to avoid them executing at the same time or having to queue. When you create a cyclic interrupt OB, the cycle time 100 and the phase offset 0 are entered as the start values.

Procedure

To enter a scan time and a phase shift for a cyclic interrupt OB, proceed as follows:

1. Open the "Program blocks" folder in the project tree.
2. Right-click on an existing cyclic interrupt OB.
3. Select the "Properties" command in the shortcut menu.

   The "<Name of the cyclic interrupt OB>" dialog box opens.
4. Click the "Cyclic interrupt" group in the area navigation.

   The text boxes for the scan time and the phase shift are displayed.
5. Enter the scan time and the phase shift.
6. Confirm your entries with "OK".

See also

Basics of block parameters (Page 581)

Organization blocks for cyclic interrupts (Page 577)

Symbolic and numerical names of instructions

Description

The instructions from the task card are comprised of functions (FC), function blocks (FB), system functions (SFC) and system function blocks (SFB) that are identified internally by numbers.

The following tables show the assignment of numerical and symbolic names.

Function blocks (FBs)

Numerical name	Symbolic name
FB 105	TC_CONFIG
FB 110	Port_Config
FB 111	Send_Config
FB 112	Receive_Config
FB 113	Send_P2P
FB 114	Receive_P2P
FB 115	Receive_Reset
FB 116	Signal_Get
FB 117	Get_Features
FB 118	Set_Features
FB 163	TC_SEND
FB 164	TC_RECV
FB 165	TC_CON
FB 166	TC_DISCON
FB 804	SET_TIMEZONE
FB 1030	TSEND_C
FB 1031	TRCV_S
FB 1071	USS_DRIVE
FB 1080	MB_COMM_LOAD
FB 1081	MB_MASTER
FB 1082	MB_SLAVE
FB 1084	MB_CLIENT
FB 1085	MB_SERVER
FB 1100	MB_Halt
FB 1101	MC_Home
FB 1102	MC_MoveAbsolute
FB 1103	MC_MoveJog
FB 1104	MC_MoveRelative
FB 1105	MC_MoveVelocity
FB 1107	MC_Power
FB 1108	MC_Reset

Numerical name	Symbolic name
FB 1110	MC_MoveInterrupt
FB 1111	MC_ChangeDynamik
FB 1112	MC_CommandTable
FB 1113	MC_MoveLinearAbs_2D
FB 1114	MC_MoveLinearRel_2D
FB 1115	MC_MoveCircular_2D
FB 1130	PID_Compact
FB 1134	PID_3Step
FB 1140	HSC
FB 2040	RecipeCreate
FB 2041	RecipeOpen
FB 2042	RecipeRead
FB 2043	RecipeWrite
FB 2044	RecipeAppend
FB 2045	RecipeClose

Functions (FCs)

Numerical name	Symbolic name
FC 2 (1)	CONCAT
FC 4 (1)	DELETE
FC 11 (1)	FIND
FC 17 (1)	INSERT
FC 20 (1)	LEFT
FC 21 (1)	LEN
FC 22 (1)	LIMIT
FC 25 (1)	MAX
FC 26 (1)	MID
FC 27 (1)	MIN
FC 31 (1)	REPLACE
FC 32 (1)	RIGHT
FC 36 (1)	ENCO
FC 36 (1)	SEL
FC 37	DECO
FC 800	LED
FC 801	IM_DATA
FC 802	DeviceStates
FC 803	ModuleStates
FC 1070	USS_PORT
FC 1072	USS_RPM

Numerical name	Symbolic name
FC 1073	USS_WPM
[1] MC7+ instruction	

System data types (SDTs)

Numerical name	Symbolic name
SDT 99	WWW_CDB
SDT 513	CONDITIONS
SDT 581	Send_Conditions
SDT 582	Receive_Conditions

System function blocks (SFBs)

Numerical name	Symbolic name
SFB 0 [1]	CTU
SFB 1 [1]	CTD
SFB 2 [1]	CTUD
SFB 3 [1]	TP
SFB 4 [1]	TON
SFB 5 [1]	TOF
SFB 27	START_OB
SFB 52	RDREC
SFB 53	WRREC
SFB 54	RALRM
SFB 105	T_CONFIG
SFB 106	TDIAG
SFB 107	TRESET
SFB 110	PORT_CFG
SFB 111	SEND_CFG
SFB 112	RCV_CFG
SFB 113	SEND_PTP
SFB 114	RCV_PTP
SFB 115	SGN_GET
SFB 116	SGN_SET
SFB 117	RCV_RST
SFB 120	CTRL_HSC
SFB 122	CTRL_PWM
SFB 140	DataLogCreate
SFB 141	DataLogOpen
SFB 142	DateLogWrite

Numerical name	Symbolic name
SFB 143	DataLogClear
SFB 144	DataLogClose
SFB 145	DataLogDelete
SFB 146	DataLogNewFile

System functions (SFCs)

Numerical name	Symbolic name
SFC 7	DP_PRAL
SFC 11	DPSYC_FR
SFC 13	DPNRM_DG
SFC 14	DPRD_DAT
SFC 16	RD_OBINF
SFC 23	DEL_DB
SFC 28	SET_TINT
SFC 29	CAN_TINT
SFC 30	ACT_TINT
SFC 31	QRY_TINT
SFC 32	SRT_DINT
SFC 33	CAN_DINT
SFC 34	QRY_DINT
SFC 41	DIS_AIRT
SFC 42	EN_AIRT
SFC 43	RE_TRIGR
SFC 45	DE_ACT
SFC 46	STP
SFC 82	CREA_DBL
SFC 83	READ_DBL
SFC 84	WRIT_DBL
SFC 86	CREATE_DB
SFC 89	RST_EVOV
SFC 99	WWW
SFC 101	RTM
SFC 117	GET_DIAG
SFC 124	ATTR_DB
SFC 140	IO2MOD
SFC 143	RD_ADDR
SFC 154	RD_LOC_T
SFC 154	DPWR_DAT
SFC 161	WR_LOC_T
SFC 180	ID2LOG

Numerical name	Symbolic name
SFC 181	LOG2ID
SFC 182	ID2GEO
SFC 190	SET_CINT
SFC 191	QRY_CINT
SFC 192	ATTACH
SFC 193	DETACH
MC7+ Anweisung	GET_ERROR
MC7+ Anweisung	GET_ERR_ID

8.1.5.2 Distributed I/O

Distributed I/O systems

SIMATIC ET 200 - The right solution for all applications

SIMATIC ET 200 provides the most varied range of distributed I/O systems.

- Solutions for use in the control cabinet
- Solutions without control cabinet directly at the machine

Additionally, there are also components that can be used in explosive areas. SIMATIC ET 200 systems for construction without a control cabinet are contained in robust, glass-fibre reinforced plastic casing and are therefore shock-resistant, resistant to dirt and watertight.

Their modular design allows the ET 200 systems to be easily scaled and expanded in small steps. Fully-integrated auxiliary modules lower costs and also provide a wide range of possible applications. There are several combination possibilities available:

- Digital and analog I/OS
- Intelligent modules with CPU functions,
- Safety technology,
- Pneumatics,
- Frequency converters
- Various technology modules.

Communication via PROFIBUS and PROFINET, uniform engineering, clear diagnostic possibilities as well as optimal connection to SIMATIC controller and HMI devices vouch for the unique consistency provided by Totally Integrated Automation.

The following table provides an overview of I/O devices for use in the control cabinet:

I/O device	Properties
ET 200S	• Highly modular design with multiple conductor connections • Multifunctional due to a wide range of modules • Use in explosive areas (Zone 2)
ET 200S COMPACT	• Highly modular design with multiple conductor connections • Multifunctional due to a wide range of modules • Use in explosive areas (Zone 2) • Integrated DE/DA
ET 200L	• Cost-effective digital block I/OS • Digital electronic blocks up to 32 channels
ET 200M	• Modular design with standard modules from SIMATIC-S7-300 • Failsafe I/O modules • Use in explosive areas up to Zone 2, sensors and actuators up to Zone 1 • High level of plant availability, for example by plugging and unplugging when in operation
ET 200iSP	• Modular design, also possible with redundancy • Robust and intrinsically safe design • Use in explosive areas up to Zone 1/21; sensors and actuators even up to Zone 0/20 • High level of plant availability, for example by plugging and unplugging when in operation

The following table provides an overview of I/O devices for use without a control cabinet:

I/O device	Properties
ET 200pro	• Modular design with compact housing • Easy assembly • Multifunctional due to a wide range of modules • High level of availability due to plugging and unplugging in operation and permanent wiring • Comprehensive diagnostics
ET 200eco PN	• Cost-efficient, space-saving block I/OS • Digital modules up to 16 channels (also configurable) • Analog modules, IO-link master and load voltage distributor • PROFINET connection with 2-port switch in each module • Can be flexibly distributed via PROFINET in line or star shape directly within the plant
ET 200eco	• Cost-effective digital block I/OS • Flexible connection possibilities • Failsafe modules • High level of plant availability
ET 200R	• Specially for use on robots • Assembled directly on the chassis • Resistant to weld spatter due to robust metal housing

See also

Documentation on ET 200L (http://support.automation.siemens.com/WW/view/de/1142908/0/en)

Documentation on ET 200S (http://support.automation.siemens.com/WW/view/en/1144348)

Documentation on ET 200M (http://support.automation.siemens.com/WW/view/de/1142798/0/en)

Documentation on ET 200pro (http://support.automation.siemens.com/WW/view/de/21210852/0/en)

Documentation on ET 200iSP (http://support.automation.siemens.com/WW/view/de/28930789/0/en)

Documentation on ET 200R (http://support.automation.siemens.com/WW/view/de/11966255/0/en)

Documentation on ET 200eco PN (http://support.automation.siemens.com/WW/view/de/29999018/0/en)

Documentation on ET 200eco (http://support.automation.siemens.com/WW/view/de/12403834/0/en)

ET 200iSP

ET 200iSP Distributed I/O Station

Definition

The ET 200iSP distributed I/O station is a highly modular and intrinsically safe DP slave with degree of protection IP 30.

Area of application

The ET 200iSP distributed I/O station can be operated in potentially explosive atmospheres characterized by gas and dust:

Approval	ET 200iSP Station*	Inputs and outputs
ATEX	Zone 1, Zone 21	up to Zone 0, Zone 20 **
IECEx	Zone 2, Zone 22	up to Zone 0, Zone 20 **
* In combination with an appropriate enclosure ** for electronic module 2 DO Relay UC60V/2A: up to Zone 1, Zone 21		

The ET 200iSP distributed I/O station can, of course, also be used in the safety area.

You can insert almost any combination of ET 200iSP I/O modules directly next to the interface module that transfers the data to the DP master. This means you can adapt the configuration to suit your on-site requirements.

Every ET 200iSP consists of a power supply module, an interface module and a maximum of 32 electronic modules (for example digital electronics modules). Remember not to exceed the maximum current consumption.

Terminal modules and electronic modules

In principle, the ET 200iSP distributed I/O station consists of various passive terminal modules onto which you plug the power supply and the electronic modules.

The ET 200iSP is connected to PROFIBUS RS 485-IS by means of a connector on terminal module TM-IM/EM. Every ET 200iSP is a DP slave on the PROFIBUS RS 485-IS.

DP master

All ET 200iSP modules support communication with DP masters that are compliant with *IEC 61784-1:2002 Ed1 CP 3/1* and operate with "DP" transmission protocol (DP stands for distributed peripherals or distributed I/O).

See also

Documentation on ET 200iSP (http://support.automation.siemens.com/WW/view/de/28930789/0/en)

Assigning the channel and IEEE tag

Properties

Analog electronic modules 4 AI I 2WIRE/HART, 4 AI I 4WIRE/ HART and 4 AO I HART support up to four IEEE tags.

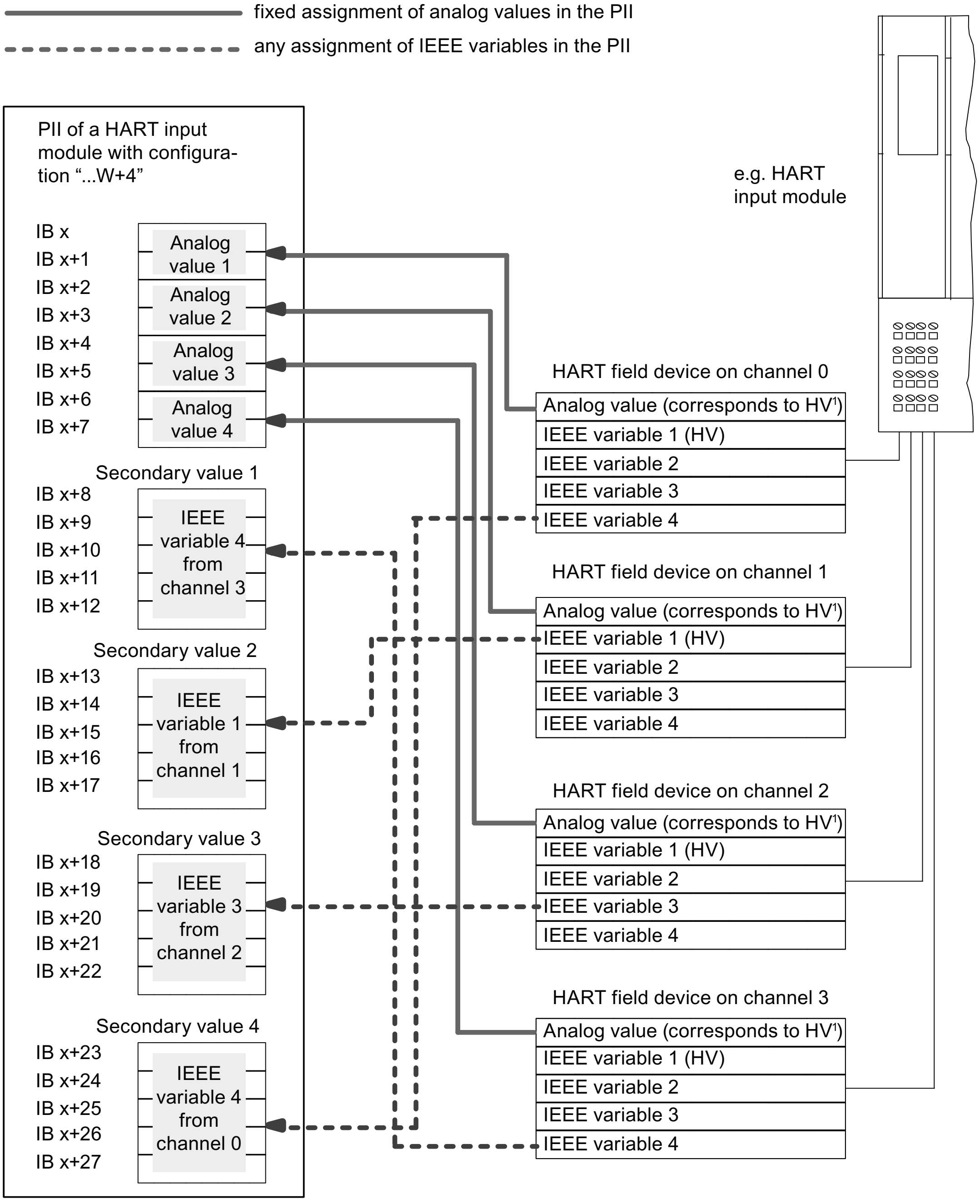
The process input image (PII) provides up to 20 bytes per module for the IEEE tags. Thus, four blocks of 5 bytes each are available for the four IEEE tags within the PII.

Requirements

The HART field device must support the assigned number of IEEE tags.

Assigning IEEE tags

You assign the IEEE tags of the field devices to any one of the four blocks in the PII.



[1] HV = main variable

See also

Documentation on ET 200iSP
(http://support.automation.siemens.com/WW/view/de/28930789/0/en)

Assigning parameters to reference junctions for thermocouples

Compensation of the reference junction temperature

There are various ways of obtaining the reference junction temperature in order to get an absolute temperature value from the temperature difference between the reference junction and the measuring point.

Table 8- 1 Compensation of the reference junction temperature

Option	Explanation	Reference junction parameters
No compensation	You record not only the temperature of the measurement point. The temperature of the reference junction (transition from Cu line to thermocouple line) also affects the thermo-electromotive force. The measured value then includes an error.	None
Use of a Pt100 Climatic Range resistance thermometer to record the reference junction temperature (best method)	You can record the reference junction temperature using a resistance thermometer (Pt100 Climatic Range). If parameterized accordingly, this temperature value is distributed to the 4 AI TC modules in the ET 200iSP where it is offset against the temperature value obtained at the measuring location. Number of reference junctions: 2	The parameter assignment of the IM 152 and the 4 AI TC must be coordinated: • 4 AI RTD assigned parameters for Pt100 climatic range in correct slot; • 4 AI TC: Reference junction : "yes"; select reference junction number "1" or "2" • IM 152-1:Assignment of the reference junction to a slot with 4 AI RTD; channel selection;
Internal compensation 4 AI TC	The TC sensor module (temperature sensor) is mounted onto the terminals of terminal module EM 4 AI TC. The temperature sensor reports the temperature of the terminals to the 4 AI TC. This value is then calculated together with the measured value from the channel of the electronic module.	• 4 AI TC: Reference junction number "internal"

Compensation by means of a resistance thermometer at the 4 AI RTD

If thermocouples that are connected to the inputs of the 4 AI RTD have the same reference junction, compensate by means of a 4 AI RTD.

For both channels of the 4 AI TC module, you can select "1", "2" or "internal" as the reference junction number. If you select "1" or "2", the same reference junction (RTD channel) is always used for all four channels.

Setting parameters for the reference junction

You set the reference junctions for the 4 AI TC electronic modules by means of the following parameters:

Table 8- 2 Reference junction parameters

Parameter	Module	Range of values	Explanation
Slot reference junction 1 to slot 2	IM 152	none, 4 to 35	With this parameter, you can assign up to 2 slots (none, 4 to 35), on which the channels for reference temperature measurement (calculating the compensation value) are located.
Input reference junction 1 to 4 input reference junction	IM 152	RTD on channel 0 RTD on channel 1 RTD on channel 2 RTD on channel 3	This parameter allows you to set the channel (0/1/2/3) for measuring the reference temperature (calculation of the compensation value) for the assigned slot.
Reference junction E0 to reference junction E3	4 AI TC	None yes	This parameter allows you to enable the use of the reference junction.
Reference junction number	4 AI TC	1 2 Internal	This parameter allows you to assign the reference junction (1, 2) that contains the reference temperature (compensation value).

See also

Documentation on ET 200iSP (http://support.automation.siemens.com/WW/view/de/28930789/0/en)

Fundamentals of Time Stamping

Properties

Time stamping is possible with the IM 152 in customer applications using FB 62 (FB TIMESTMP).

Principle of operation

A modified input signal is assigned a time stamp and stored in a buffer (data record). If time stamped signals exists or a data record is full, a hardware interrupt is generated to the DP master. The buffer is evaluated with "Read data record". Special messages are generated for events that influence the time stamping (communication with the DP master interrupted, frame failure of time master, ...).

Parameter Assignment

With the parameter assignment you define which IM 152 user data will be monitored. For the time stamping these are digital inputs that are monitoring for signal changes.

Parameter	Setting	Description
Time stamping	• disabled • enabled	Activate the time staming for the channels of the electronics module 8 DI NAMUR.
Edge evaluation incoming event	• rising edge • falling edge	Determine the type of signal change that will be time-stamped.

Counting

Count properties

Counting functions

The 8 DI NAMUR electronics module has configurable counting functions:

- 2 x 16-bit up counters (standard counting function) or
- 2 x 16-bit down counters (standard counting function) or
- 1 x 32-bit down counter (cascading counter function)
- Setting a setpoint with the PIQ
- GATE function
- You can configure the control signals of the counters:
  - Configuration channel 0..1: "Counter", channel 2..7: "DI": Two counters are configured. The control signals of the counters are stored in the PIQ (process image output).
  - Configuration channel 0..1: "Counter", channel 2..7: "Control": Two counters are configured. The control signals of the counters are stored in the PIQ (process image output). They are also controlled by the digital inputs of the 8 DI NAMUR.

See also

Principle of operation (Page 597)

Configuring counters (Page 599)

Assigning parameters to counters (Page 602)

Principle of operation

16-bit up counters (standard counting function)

The counting range is 0 to 65,535.

With each count pulse at the digital input, the count is incremented by 1. Once the count limit is reached, the counter is reset to 0 and it counts up again from this value.

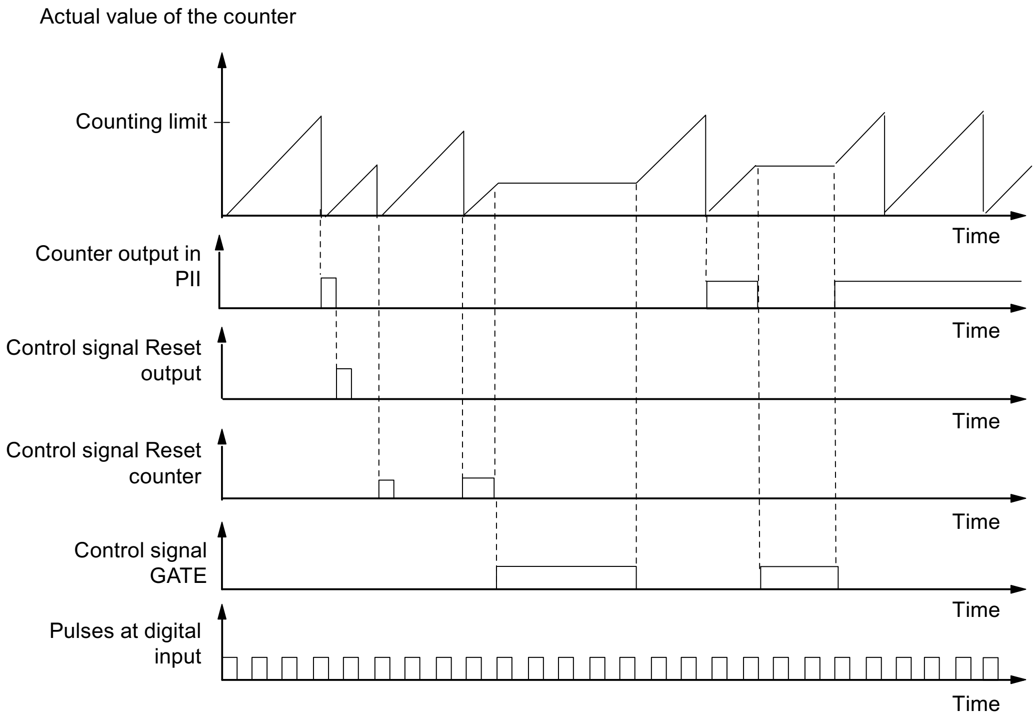
If there is counter overflow, the corresponding output is set in the PII.

A positive edge of the *Reset output* control signal resets the output in the PII. This does not affect the current count value.

In 16-bit up counting operations, the system does not set any outputs in the PIQ. These are always reset.

The positive edge of the *Reset counter* control signal sets the counter to 0 and resets the set counter output.

The *GATE* control signal pauses the counting on a positive edge. Count pulses are processed at the digital input again, but only at the negative edge. The *Reset counter* control signal is also effective when *GATE* is active.



16-bit down counters (periodic counting function)

The maximum counting range is always 65,535 to 0.

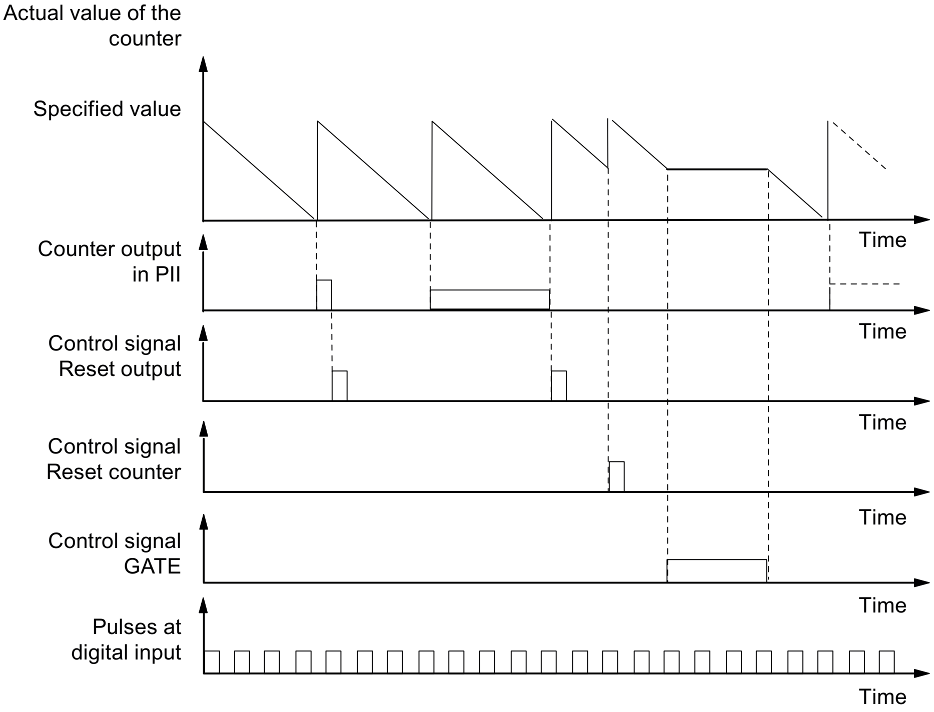
When the counter is started, the actual value is set to the selected setpoint. Each counted pulse reduced the actual value by 1. Once the actual value reaches 0, the corresponding output in the PII is turned on and the actual value is set to the selected setpoint. The counter then counts down from this value.

The positive edge of the *Reset counter* control signal resets the selected setpoint and the corresponding output in the PII.

A positive edge of the *Reset output* control signal resets the output in the PII. This does not affect the current count value.

The *GATE* control signal pauses the counting on a positive edge. At the same time, the assigned output in the PII is reset. Count pulses are processed at the digital input again, but only at the negative edge. The *Reset output* and *Reset counter* control signals are also effective when *GATE* is active.

The setpoint of the counter is set and changed using the PIQ. The setpoint is adopted on a positive edge of the *Reset counter* control signal or when the counter has reached zero.



32-bit down counter (cascading counter function)

The maximum counting range is always 4294967295 to 0.

The principle of operation is identical to that of the 16-bit down counter. Channel 1 has no function.

See also

Count properties (Page 596)

Configuring counters

Procedure

1. Using the mouse, pull module 8 DI Namur from the hardware catalog into distributed I/O station ET 200iSP.
2. Select the required configuration (channel 0..1: "Counter", channel 2..7: "DI" or "Control"). In the module properties (inspector window), you can find this setting under "Parameters > Inputs > Configuration".

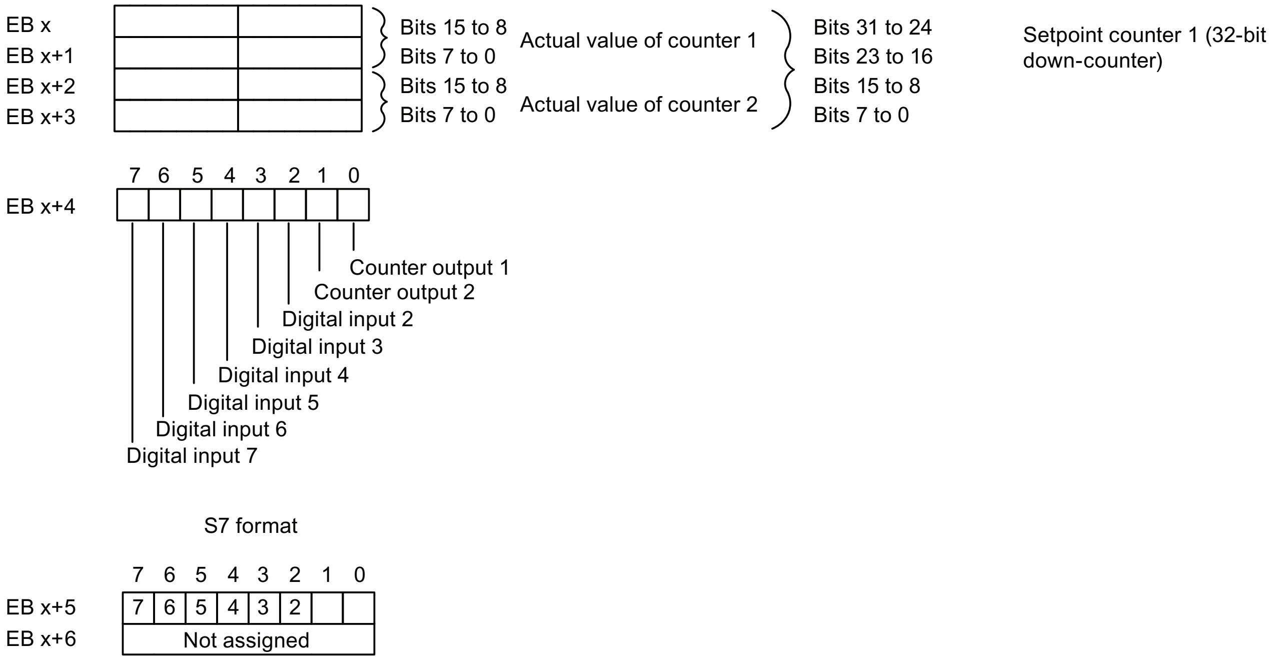
Configuration channel 0..1: "Counter", channel 2..7: "DI"

- Assignment of the digital inputs on the electronic module 8 DI NAMUR

Table 8- 3 Assignment of digital inputs for channel 0..1: "Counter", channel 2..7: "DI":

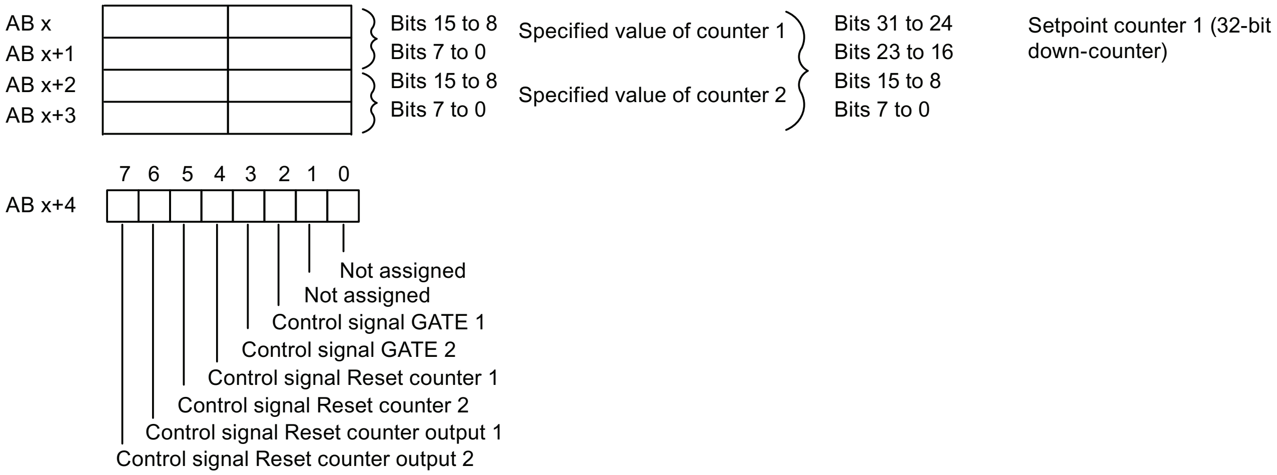
Digital input	Terminal	Assignment
Channel 0	1, 2	Counter 1
Channel 1	5, 6	Counter 2 (does not apply to 32-bit down counters)
Channel 2	9, 10	Digital input 2
Channel 3	13, 14	Digital input 3
Channel 4	3, 4	Digital input 4
Channel 5	7, 8	Digital input 5
Channel 6	11, 12	Digital input 6
Channel 7	15, 16	Digital input 7

- Assignment of the process image input (PII)



Value status for channels 2 to 7:
1_B: Input signal is valid
0_B: Input signal is invalid

- Assignment of the process image output (PIQ)


AB x
AB x+1
AB x+2
AB x+3
Bits 15 to 8
Bits 7 to 0
Specified value of counter 1
Bits 15 to 8
Bits 7 to 0
Specified value of counter 2
Bits 31 to 24
Bits 23 to 16
Bits 15 to 8
Bits 7 to 0
Setpoint counter 1 (32-bit down-counter)
7 6 5 4 3 2 1 0
AB x+4
Not assigned
Not assigned
Control signal GATE 1
Control signal GATE 2
Control signal Reset counter 1
Control signal Reset counter 2
Control signal Reset counter output 1
Control signal Reset counter output 2

Configuration channel 0..1: "Counter", channel 2..7: "CONTROL"

With this configuration, you can also control the counters over the digital inputs.

- Assignment of the digital inputs on electronic module 8 DI NAMUR

  For further information on input assignments, refer to the technical data for electronic module 8 DI NAMUR.

Table 8- 4 Assignment of the digital inputs for 2 Count/ 6 Control

Digital input	Terminal	Assignment
Channel 0	1, 2	Counter 1
Channel 1	5, 6	Counter 2 (does not apply to 32-bit down counters)
Channel 2	9, 10	control signal *GATE 1*
Channel 3	13, 14	control signal *GATE 2*
Channel 4	3, 4	control signal *Reset counter 1*
Channel 5	7, 8	control signal *Reset counter 2*
Channel 6	11, 12	control signal *Reset counter output 1*
Channel 7	15, 16	control signal *Reset counter output 2*

- Assignment of the process image input (PII)

  Assignment is identical to configuration 0..1: "Counter", channel 2..7: "DI".

- Assignment of the process image output (PIQ)

  Assignment is identical to configuration 0..1: "Counter", channel 2..7: "DI".

See also

Count properties (Page 596)

Assigning parameters to counters

Parameters for the counting function

Only those parameters that are relevant for the counters are explained below. These belong to the parameters of electronic module 8 DI NAMUR and depend on the selected configuration:

Table 8- 5 Parameters for the counters

Parameter	Setting	Description
Sensor type counter inputs	• Channel disabled • NAMUR sensor • Single contact, no load resistance	Select the sensor for the respective counter of channels 0 or 1.
Mode for counter 1	• Standard counting function • Periodic counting function • Cascaded counting function	Select the mode for counter 1.
Mode for counter 2	• Standard counting function • Periodic counting function • Cascaded counting function	Select the mode for counter 2. This parameter is not relevant if you have set the "Mode for counter 1" parameter to "Cascaded counter function".

See also

Count properties (Page 596)

Frequency measurement

Frequency measurement properties

Properties

The electronic module 8 DI NAMUR allows the frequencies to be measured on channel 0 and 1:

- 2 frequency meters from 1 Hz to 5 kHz
- Configurable metering window (GATE)
- The signals of the frequency meter are read in by means of the digital inputs of the electronic module.

See also

Principle of operation (Page 603)

Configuring frequency meters (Page 603)

Assigning parameters for the frequency meters (Page 605)

Principle of operation

Frequency measurement

The signal frequencies are identified from the input signals of channel 0 or 1 of the electronic module. To calculate the frequency the signals are measured within a configurable gate.

The frequency is displayed as 16-bit value in fixed-point format and transferred to the PII.

The frequency meter calculates the frequency according to the follow formula:

$$\text{Frequency [Hz]} = \frac{\text{Number of rising edges at digital input}}{\text{Measuring window [s]}}$$

Exceeding the input frequency

If the input frequency exceeds 5kHz, $7FFF_H$ is reported as actual value. If the input frequency is above approx. 8 kHz it is no longer possible to display correct actual values.

See also

Frequency measurement properties (Page 602)

Configuring frequency meters

Procedure

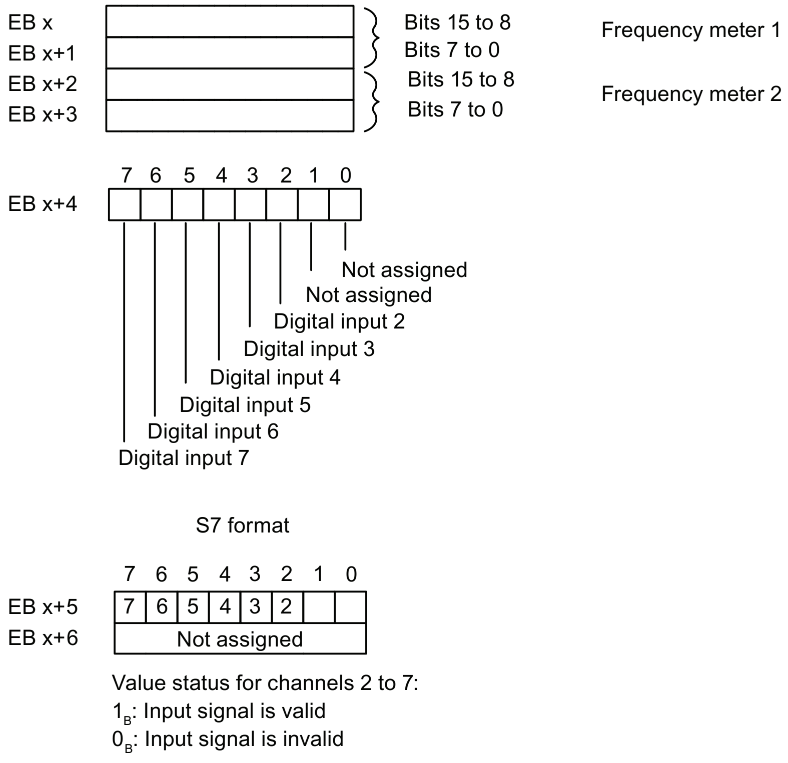
1. Using the mouse, pull module 8 DI Namur from the hardware catalog into distributed I/O station ET 200iSP.
2. Select the required configuration (channel 0..1: "Trace", channel 2..7: "DI"). In the module properties (inspector window), you can find this setting under "Parameters > Inputs > Configuration".

Configuration 0..1: "Trace", channel 2..7: "DI"

Assignment of the digital inputs on electronic module 8 DI NAMUR

Digital input	Terminal	Assignment
Channel 0	1, 2	Frequency counter 1
Channel 1	5, 6	Frequency counter 2
Channel 2	9, 10	Digital input 2
Channel 3	13, 14	Digital input 3
Channel 4	3, 4	Digital input 4
Channel 5	7, 8	Digital input 5
Channel 6	11, 12	Digital input 6
Channel 7	15, 16	Digital input 7

Assignment of process image input (PII) for configuration of channel 0..1: "Trace", channel 2..7: "DI"



Assignment of the process image output (PIQ): The PIQ is not assigned.

See also

Frequency measurement properties (Page 602)

Assigning parameters for the frequency meters

Parameters for frequency meter

Only those parameters that are relevant for the frequency meters are explained below. These are part of the parameters of electronic module 8 DI NAMUR.

Table 8- 6 Parameters for the frequency meters

Parameter	Setting	Description
Sensor type frequency inputs	• Channel disabled • NAMUR sensor • Single contact, no load resistance	Select the sensor for the relevant frequency meter for channel 0 or 1.
Measuring window (GATE)	• 50 ms • 200 ms • 1 s	Select the required measuring window for channel 0 or 1. To achieve the highest possible accuracy when metering frequencies, remember the following rules: • High frequencies (> 4 kHz): Set a low measuring window (50 ms) • Variable/medium frequencies: set medium measuring window (200 ms) • Low frequencies (< 1 kHz): Set a high measuring window (1 s)

See also

Frequency measurement properties (Page 602)

ET 200eco PN

ET 200eco PN Distributed I/O Device

Definition

The ET 200eco PN distributed I/O device is a compact PROFINET IO device in degree of protection IP 65/66 or IP 67 and UL Enclosure Type 4x, Indoor use only.

Field of application

The fields of application of the ET 200eco PN are derived from its special properties.

- A robust design and degree of protection IP 65/66 or IP 67 make the ET 200eco PN distributed I/O device suitable in particular for use in rugged industrial environments.
- The compact design of the ET 200eco PN is particularly favorable for applications in confined areas.
- The easy handling of ET 200eco PN facilitates efficient commissioning and maintenance.

Properties

The ET 200eco PN has the following properties:

- Integrated switch with 2 ports
- Supported Ethernet services:
  - ping
  - arp
  - Network diagnostics (SNMP)
  - LLDP
- Interrupts
  - Diagnostics interrupts
  - Maintenance interrupts
- Port diagnostics
- Isochronous real-time communication
- Prioritized startup
- Device replacement without programming device
- Media redundancy
- Connection to intelligent sensors/actuators via IO link master interface module.

IO Controller

The ET 200eco PN can communicate with all IO Controllers that conform to IEC 61158.

ET 200eco PN can be configured on a CPU with advanced diagnostics.

See also

Documentation on ET 200eco PN
(http://support.automation.siemens.com/WW/view/en/29999018)

Parameter description analog input

Group diagnostics

You can generally enable and disable the diagnostics function of the device with this parameter.

The "Fault" and "Parameter assignment error" diagnostics functions are always independent of the group diagnostics.

Diagnostics, missing 1L+

If you enable this parameter, the check for missing supply voltage is enabled.

Diagnostics, sensor supply short circuit

If you enable this parameter, a diagnostics event is generated if a short-circuit of the sensor supply to ground is detected and the channel is enabled. The sensor supply is monitored for connectors X1, X3, X5 and X7. It is not possible to differentiate which connector has experienced the sensor short circuit.

Interference frequency suppression

With this parameter, you set the integration time of the device, based on the selected interference frequency. Select the frequency of the supply voltage used. Interference frequency suppression Off means 500 Hz, which corresponds to an integration time of 2 ms for a measurement channel.

Temperature unit

Specify the unit of the temperature measurement here.

Measurement type (channel-wise)

With this parameter, you set the measurement type, for example, voltage. For any unused channels, you must select the disabled setting. For a disabled channel, the conversion time and integration time of the channel = 0 s and the cycle time is optimized.

Measuring range

With this parameter, you set the measuring range of the selected measurement type.

Temperature coefficient (for RTD, thermoresistor)

The correction factor for the temperature coefficients (α-value) indicates by what extent the resistance of specific material changes relatively if the temperature increases by 1 °C.

The α-values conform to EN 60751, GOST 6651, JIS C 1604 and ASTM E-1137.

The temperature coefficient depends on the chemical composition of the material.

Smoothing

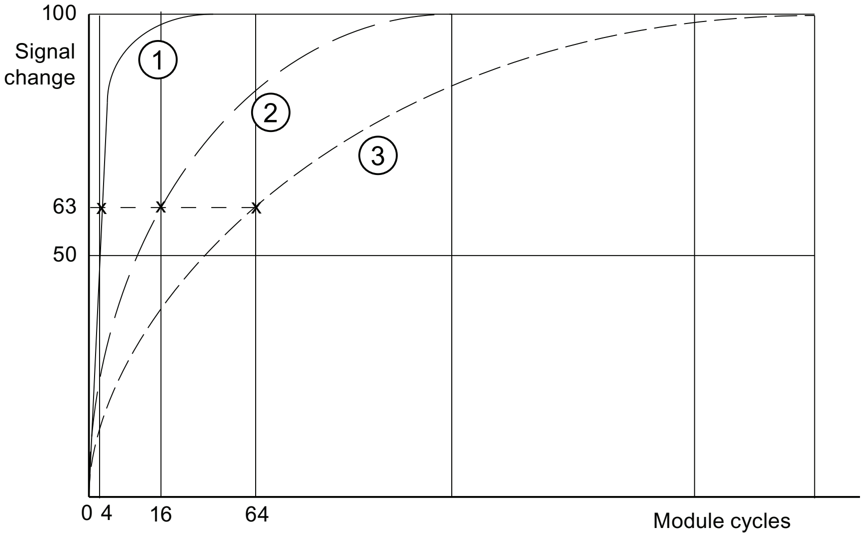
Smoothing of the analog values produces a stable analog signal for further processing. The smoothing of analog values is useful when handling wanted signals (measured values) with a slow rate of change, for example, temperature measurements.

The measured values are smoothed by digital filtering. To achieve smoothing, the device generates a mean value from a specified number of converted (digitized) analog values.

You assign a maximum of four levels for the smoothing (none, weak, medium, strong). The level determines the number of module cycles, from which the mean value is generated.

The stronger the smoothing, the more stable the smoothed analog value and the longer it takes until the smoothed analog value is applied following a signal change (see the example below).

The figure below shows the number of cycles a module requires to apply the smoothed analog value at almost 100% after a step response, based on the smoothing function settings. The figure applies to all signal changes at the analog input. The smoothing value defines the number of cycles a module requires to reach 63% of the end value of the changed signal.



① Smoothing, weak

② Smoothing, medium

③ Smoothing, strong

Diagnostics, wire break

When this parameter is enabled, the Wire break diagnostics event is generated when a wire break is detected.

Observe the rules outlined below to handle a wire break in the 1 to 5 V and 4 to 20 mA measuring ranges:

Parameter	Event	Measured value	Explanation
Enable wire break[1]	Wire break	$7FFF_H$	Diagnostics, **wire break**
Wire break disabled[1] Underflow enabled	Wire break	8000_H	Measured value after leaving the underrange Diagnostic message **Lower limit value undershot**
Wire break disabled[1] Underflow disabled	Wire break	8000_H	Measured value after leaving the underrange

[1] Measuring range limits for wire break detection and measuring range undershoot detection:

- 1 to 5 V: At 0.296 V
- 4 to 20 mA: At 1.185 mA

Diagnostics, underflow

If you enable this parameter, the Underflow diagnostics event is generated when the measured value reaches the underflow range.

Diagnostics, overflow

If you enable this parameter, the Overflow diagnostics event is generated when the measured value reaches the overflow range.

Reference junction for thermoresistor (TC)

A difference in temperature between the measuring point and the free ends of the thermocouple (terminal point) generates a voltage between the free ends, namely the thermoelectric voltage. The value of this thermoelectric voltage is determined by the temperature difference between the measuring point and the free ends and by the type of material combination of the thermocouple. Since a thermocouple always measures a temperature difference, the free ends at the reference junction must be maintained at a known temperature in order to determine the temperature of the measuring point.

If you specify Internal compensation, the temperature of the measuring point in the housing of the I/O device is measured. With the External compensation setting, you can connect a compensation box in series in order to increase the accuracy of the temperature measurement.

Parameter description analog output

Group diagnostics

You can generally enable and disable the diagnostics function of the device with this parameter.

The "Fault" and "Parameter assignment error" diagnostics functions are always independent of the group diagnostics.

Diagnostics, missing 1L+

If you enable this parameter, the check for missing supply voltage is enabled.

Diagnostics, sensor supply short circuit

When this parameter is enabled, the system generates a diagnostics event if it detects a short-circuit of the sensor supply to ground. This diagnostics function is activated when the group diagnostics function is enabled.

Response to CPU/Master STOP

Select how the module's outputs will respond to a CPU STOP:

- Shut down

  The I/O device goes to the safe state. The process image output is deleted (=0).

- Keep last value

  The I/O device retains the last value to be output before STOP.

- Substitute value

  The I/O device outputs the value for the channel set beforehand.

NOTICE
Make sure that the plant is always in a safe state if "Keep last value" is selected.

Type of output

With this parameter, you set the output type, for example, voltage. For any unused channels, select the disabled setting. For a disabled channel, the conversion time and integration time of the channel = 0 s, and the cycle time is optimized.

Output range

With this parameter, you set the output range of the selected output type.

Diagnostics, wire break (in current mode)

When this parameter is enabled, the **Wire break** diagnostics event is generated when a wire break is detected. This diagnostics event cannot be detected in the zero range.

Diagnostics, short circuit (in voltage mode)

If you enable this parameter, a diagnostics event is generated in the event of a short circuit in the output line. This diagnostics event cannot be detected in the zero range.

Diagnostics, overload

If you enable this parameter, the diagnostics event is generated in the event of an overload.

Substitute values

With this parameter, you enter a substitute value that the module is to output in CPU-STOP mode. The substitute value must be in the nominal range, overrange, or underrange.

Signal modules for process automation

Fundamentals

Introduction

Signal modules for the process automation are S7-300 models, such as SM 321; DI 16xNAMUR or SM 322; DO 16x24VDC/0.5A.

They are being operated in a DP slave (IM 153-2).

Unlike standard modules, they offer the following additional technical functions, such as pulse extension and chatter monitoring.

See also

Changeover contact (Page 611)

Technological parameters (Page 612)

Changeover contact

"Changeover contact" sensor type

If the digital inputs of a channel group are configured as "changeover contacts", the module runs diagnostics for the changeover contact sensor type for this channel group.

Changeover contact

A changeover contact is an auxiliary switch with only one moving switch element with one close setting each for closed and open switching device.

Remember the following rule:

- Always connect a normally open contact to the "even" channel
- Always connect a normally closed contact to the "odd" channel.

The tolerated switchover time between the two channels is fixed at 300 ms.

If the result of the check is negative, then

- the module identifies the value status of the normally open channel as "invalid"
- the module generates a diagnostic entry for the normally open channel
- triggers a diagnostic interrupt (if diagnostic interrupts have been enabled)

The digital input signal and the value status are updated only for the normally open channel. For the normally closed channel, the digital input signal is set permanently to "zero" and the value status to "invalid" since this channel is used only to check the sensor.

Diagnostics depends on the "Selection" parameter (of the sensor). You should also note the special features of diagnostics with the changeover contact sensor type in the "Signal Modules for Process Automation" manual.

See also

Documentation on modules for process automation (http://support.automation.siemens.com/WW/view/de/7215812/0/en)

Technological parameters

Pulse extension and flutter monitoring

Pulse extension is a function for changing a digital input signal. A pulse at a digital input is extended to at least the length set in the parameters. If the input pulse is already longer than the specified length, it is not changed.

If you want the pulse to be extended, click in the box to select the time. If you do not want the pulses to be extended, select the "---" entry.

Flutter monitoring is a process control function for digital input signals. It detects and reports signal changes that are unexpected in process control, for example when an input signal fluctuates too often between "0" and "1".

Flutter monitoring is possible only when group diagnostics has also been enabled for this input.

Monitoring window and number of signal changes

Flutter monitoring works with aid of the two parameters Monitoring window and Number of signal changes.

The first time the signal changes, the time set as the monitoring window is started. If the signal changes more often during this time than allowed by the number of signal changes parameter, this is signaled as a flutter error. If no flutter error is detected during the monitoring window time, the monitoring window can be restarted at the next signal change.

Note

If you set pulse extension for an input channel, this also affects the flutter monitoring enabled for this channel. The "extended pulse" signal is the input signal for the flutter monitoring. You should therefore make sure that the values set for pulse extension and flutter monitoring are compatible with each other.

See also

Documentation on modules for process automation (http://support.automation.siemens.com/WW/view/de/7215812/0/en)

Frequency converters

Use of the frequency converter

Frequency converters

The frequency converter ICU24 and ICU24F (as fail-safe version) are modular design frequency converters that are completely embedded in the distributed I/O system ET 200S. For parameterization of both modules, please see the following.

Message frame

The message frame number and the operating mode of the module are only displayed and cannot be modified.

Application ID

You indicate the saved parameters in the frequency converter as a whole with the application ID. Enter an application ID from the value range 0 to 65535. During startup (or pull/plug), this ID is compared with the application ID stored on the converter.

Converters that work with identical applications are usually also identically parameterized and should be identified with the same application ID. Converters with the same application ID may be exchanged between each other. Copying of the complete parameterization of a converter to another converter, for example, via an MMC, is only accepted, if both have the same application ID.

Converters that work with different applications and are parameterized differently must be identified by different application IDs. This prevents a converter with unsuitable parameterization from starting on an incorrect slot, i.e. on the wrong application. This also prevents the parameterization that is saved in the converter from being accidently overwritten with any parameteriation that is stored on an MMC.

Enable diagnostic interrupt

You can enable the diagnostic interrupt for the frequency converter. If diagnostic interrupt is enabled, an OB 82 must be available in a CPU to process the diagnostic events.

See also

Documentation for the frequency converter
(http://support.automation.siemens.com/WW/view/en/26291825/0/en)

IQ sense module

Properties of 4 IQ-SENSE

Properties

The 8 IQ-SENSE module has the following properties:

- Connection of sensors with IQ-SENSE®, photoelectric proximity switches: for example, reflex sensors, diffuse sensors, and laser sensors.
- It can be used centrally in an S7-300 or distributed in an ET 200M.
- You can connect up to 8 sensors to every module. Each sensor requires a two-wire cable.
- Function reserve that can be assigned parameters.
- Time functions, switching hysteresis, synchronous mode that can be assigned parameters
- Sensitivity and distance values can be specified (*IntelliTeach* using the "IQ-SENSE Opto" FB)
- Teach-in
- Sensors can be removed and inserted during operation (automatic reassignment of parameters)

See also

Sensor type (Page 615)

Switching hysteresis (Page 616)

Time function,time value (Page 617)

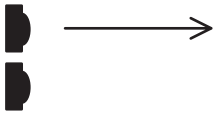
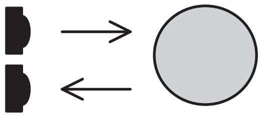
Sensor type

This parameter is used to set the sensor type per channel:

- Reflex sensor or
- Diffuse sensor or
- De-activated

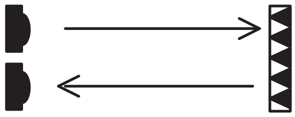
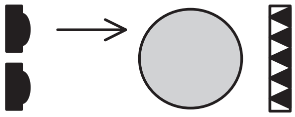
Diffuse sensor

Table 8- 7 Diffuse sensor

Diffuse sensor	Object	
Transmitter Receiver		Circuit state 0: no object detected, i.e. the object is not in the beam. The receiver does not see any light.
Transmitter Receiver		Circuit state 1: object detected, i.e. the object is in the beam. The receiver does not see any light.

Reflex sensor

Table 8- 8 Reflex sensor

Reflex sensor	Object	
Transmitter Receiver		Circuit state 0: no object detected, i.e. the object is not in the beam. The receiver sees light.
Transmitter Receiver		Circuit state 1: object detected, i.e. the object is in the beam. The receiver does not see any light.

Switching hysteresis

Faults with the diffuse sensor or in the production process can result in signal wobbles. The measured value then changes the switching threshold by 100 % (object detected - object not detected). You can prevent this switching threshold wobble using the switching hysteresis parameter. This will ensure a stable output signal on the sensor.

You can assigned parameters to 5 %/10 %/20 %/50 % for switching hysteresis.

Prerequisites

You can only set the switching hysteresis parameter for diffuse sensors with background fadeout.

Operating principle

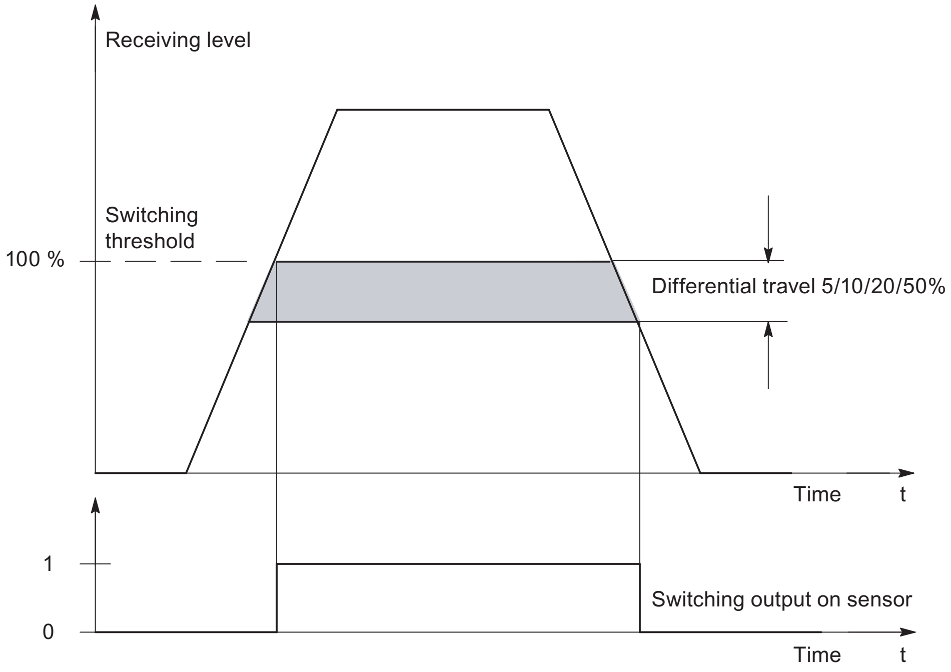


Figure 8-3 Switching hysteresis parameter

Time function,time value

These parameters can be used to set the electronic module for its specific application.

Operating principle

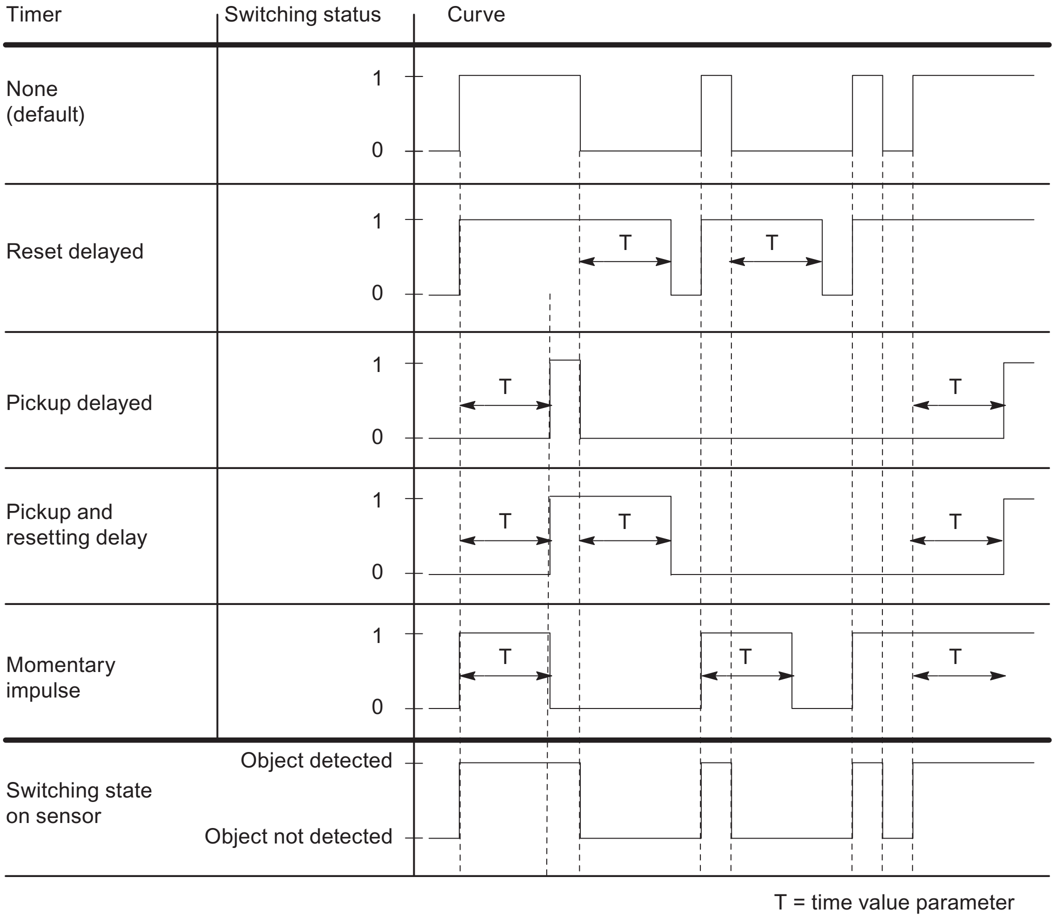


Figure 8-4 Time functions, time values parameters

8.1.5.3 Parameters of analog modules

Measurement types with thermocouples

Differences in the measurement types with thermocouples

- Thermocouples with reference junction

  TC-IL (internal reference junction, with linearization): Linearization of the characteristic and temperature compensation with the temperature of the internal reference junction (module clamping point) as the reference temperature.

  TC-EL (external reference junction, with linearization): Linearization of the characteristic and temperature compensation with the temperature of the external reference junction as the reference temperature.

- Thermocouples with reference temperature

  TC-L00C (0 ºC reference junction, with linearization): Linearization of the characteristic and temperature compensation with 0 °C / 32 °F

  TC-L50C (50 ºC reference junction, with linearization): Linearization of the characteristic and temperature compensation with 50 °C / 122 °F

Interference frequency suppression and integration time

Interference frequency suppression and integration time

The following table illustrates the interaction:

Interference frequency suppression (Hz)	Integration time (ms)
60	16,6
50	20

With some modules, you can set other frequencies for the interference frequency suppression:

Interference frequency suppression (Hz)	Integration time (ms)
10	100
400	2,5

8.1.6 System diagnostics with 'Report System Errors'

8.1.6.1 Introduction to system diagnostics with 'Report System Errors'

Introduction

When a system error occurs, hardware components and devices of other manufacturers (slaves whose properties are set by the GSD files) can trigger organization block calls.

Example: If a wire breaks, a module with diagnostics capability can call OB 82.

The hardware components provide information on the system error that has occurred. The start event information, in other words, the local data of the assigned OB (including data record 0), provides general information on the location (for example, logical address of the module) and type (for example channel error or module fault) of the error.

The problem can also be specified in greater detail using additional diagnostics information (reading data record 1 with the "RDSYSST" instruction or reading the diagnostics frame from standard DP slaves with the "DPNRM_DG" instruction): For example, channel 0 or 1, broken wire or measuring range exceeded.

System diagnostics with "Report System Errors" (RSE) provides you with a convenient way of evaluating this diagnostics information for S7-300/400 PLCs, ET200S, ET200Pro and software PLCs and to display it in the form of messages. The required blocks and alarm texts are created in the properties of the particular PLC. You only need to download the generated blocks to the CPU and, if required, transfer the texts to connected HMI devices.

To display diagnostics events graphically, for example on an HMI device or via a Web server, you can generate one or more status DBs. These status DBs are updated by the system diagnostics blocks and then contain information on the current state of the system.

8.1.6.2 Basics of system diagnostics

Using system diagnostics with "Report System Errors", you can generate blocks that analyze errors in the system and generate alarms with a textual error description and the error location.

These alarms are defined per component with alarm capabilities (for example channel errors or rack errors) and are limited to 255 alarms component with alarm capability. You will receive a message if this limit is exceeded.

General settings that you make or change in system diagnostics are stored along with the project. The influence on the system diagnostics only takes effect after generating the blocks, compiling the hardware configuration and downloading the configuration to the components involved.

Recommended procedure

Make the settings for reporting system errors and the structure of the alarms. Specify which diagnostics blocks will be created. Configure the OBs, Status DBs, as well as the PLC in STOP. Generate the blocks and download the configuration to apply the changes.

You will find detailed information in the sections below.

Note

If you use system diagnostics, the system response of the plant may change if an error occurs. For example, the CPU may not change to "STOP" mode as it would without system diagnostics.

Make sure that all protective mechanisms of the plant are working properly.

8.1.6.3 Components supported

The following components are supported:

- S7-300 CPUs
- S7-400 CPUs
- S7-400 power supply modules
- PN/PN Coupler
- DP/DP Coupler
- IE/PB-Link
- AS-i CP
- ET 200S
- ET 200M
- ET 200eco
- ET 200R
- ET 200Pro
- ET 200L
- Diagnostics repeater
- GSD-based slaves
- GSDML-based slaves
- ET 200iSP
- IO-Link
- Software CPUs

8.1.6.4 Diagnostics blocks for reporting system errors

Once you have made the settings for reporting system errors, the required blocks (FB with assigned instance DB and one or more global DBs, an FC and, depending on the setting, also OBs that do not yet exist) and status DBs are created the next time you compile the hardware configuration.

The following blocks are created:

- Diagnostics FB (default: FB 49),
- Instance DB for the diagnostics FB (default: FB 49),
- Global DB (default: FB 50),
- Diagnostics FC (default: FC 49),
- Error OBs (if you have selected this option in the "OB configuration" group box),
- Cyclic OBs (if you have selected this option in the "OB configuration" group box),
- Optional status DBs

The created blocks (except OBs) are stored in the project tree under "Program blocks > System blocks > Report System Errors".

See also

Properties of the blocks (Page 622)

8.1.6.5 Properties of the blocks

Diagnostics blocks

The created diagnostics blocks (FB with assigned instance DB and one or more global DBs and an FC) evaluate the local data of the error OB and read any additional diagnostics information from the hardware component that triggered the error.

They have the following properties:

- Created in RSE language (report system errors) (also applies to the blocks listed above)
- Know-how protected (also applies to the blocks listed above)
- Delay incoming interrupts during runtime

Status DBs

The Status DBs are used as the interface for diagnostic blocks and make it possible to

See also

Settings for system diagnostics blocks (Page 635)

8.1.6.6 Supported error OBs

The following error OBs are created if they are also supported by the configured CPU:

OB	Diagnostics FB call	Description
OB 70 (IO redundancy error)	possible	This OB exists only with H-CPUs.
OB 72 (CPU redundancy error)	possible	This OB exists only with H-CPUs.
OB 73 (communications redundancy error)	possible	This OB exists only with a few H-CPUs.
OB 81 (power supply error)	possible	
OB 82 (diagnostics interrupt OB)	possible	
OB 83 (pull/plug interrupt)	possible	
OB 85 (program execution error)	not possible	If 'RSE' creates this OB when it is generating the diagnostics blocks, additional networks are inserted to implement the following program sequence: • If errors occur when updating the process image (for example removing the module), the CPU is prevented from changing to STOP so that the diagnostic FB can be run in OB 83. If there is a setting for "CPU STOP" after an alarm from "RSE", this takes effect in OB 83. • With all other error events of OB 85, the CPU changes to STOP.
OB 86 (failure of an expansion rack, a DP master system or a distributed I/O device)	possible	

If the OBs already exist...

Existing error OBs on not overwritten. If appropriate, the call for the diagnostics FB is appended in the existing OB.

If the configuration includes distributed I/O devices...

To evaluate errors in the distributed I/O, the created FB automatically calls the "DPNRM_DG" instruction (read diagnostics data of the DP slaves). For this function to operate correctly, the created FB must be called either only in OB 1 or in a cyclic interrupt OB with a short call interval and in the startup OBs.

NOTICE

Please note the following:

- Due to the OB 85 created by "RSE", the CPU does not change to STOP if the "error updating the process image" error event occurs.
- OB 85 is also called by the CPU in the following error situations.
  - "Error event for an OB that is not loaded"
  - "Error calling or accessing a block that is not loaded"

    In these error situations, the OB 85 generated by "RSE" still changes the CPU to STOP in the same way as prior to the use of "RSE".
- The setting "PLC in STOP" does not have any effect in OB 85, because this OB of the "RSE" FB is not called. This setting is taken into account indirectly by the FB call in OB 83.

See also

Settings for OB configuration (Page 637)

8.1.6.7 Overview of the status DBs

Within system diagnostics, you have the option of creating status DBs. These status DBs are updated by the system diagnostics blocks and then contain information on the current state of the system. The following status DBs are available:

- The diagnostics status DB (default: DB 127) displays the state of racks, and central modules, PROFIBUS slaves and IO devices.
- The PROFINET IO DB (default: DB 126) displays the state of IO devices in I/O systems and the state of DP slaves in DP master systems downstream from an IE/PB-Link.
- The PROFIBUS DP DB (default: DB 125) displays the status of PROFIBUS slaves in DP master systems.

8.1.6.8 Diagnostics status DB

Interface for the diagnostics status DB

The created data block (default: DB 127) allows you to query the system state of a configured component and, if required, all its lower-level components.

This data block is required to support system diagnostics using the Web server or diagnostics viewer. As default, it is enabled.

NOTICE
Please note that system diagnostics must be activated both on the Web server or diagnostics viewer and in the Inspector window of the PLC to be able to display diagnostics data of the status DBs.

After restarting a CPU with Web server, the module state is displayed following a delay. To shorten the waiting time, you can call the RSE diagnostics block in a cyclic interrupt OB.

Structure of the diagnostics status DB

Address	Name	Data type	Description
+0	Directory		
0	D_Version	WORD	Version supported by system diagnostics
2	D_pGlobalState	WORD	Byte offset to the start of the "GlobalState" section
4	D_pQuery	WORD	Byte offset to the start of the "Query" section
6	D_pComponent	WORD	Byte offset to the start of the "Component" section
8	D_pError	WORD	Byte offset to the start of the "Error" section
10	D_pState	WORD	Byte offset to the start of the "State" section
12	D_pAlarm	WORD	Byte offset to the start of the "Alarm" section
14	D_pSubComponent	WORD	Byte offset to the start of the "Subcomponent" section
+16	GlobalState		
0	G_EventCount	WORD	ID of the last event (counter)
2.0	G_StartReporting	BOOL	Startup evaluation active
+20	Query		
0	Q_ClientID_User	DWORD	ID of the client; here, please use a value between 1 and 255. Make sure that different clients use different IDs.
4	Q_ClientID_Intern	DWORD	ID of the client (internal)
8.0	Q_WithSubComponent	BOOL	With/without status of the lower-level components (slower)
8.1	Q_SubComponentAlarm	BOOL	AS-i master returns AS-i slave interrupts
8.2	Q_Reserved2	BOOL	Reserved

Address	Name	Data type	Description
8.3	Q_Reserved3	BOOL	Reserved
8.4	Q_Reserved4	BOOL	Reserved
8.5	Q_Reserved5	BOOL	Reserved
8.6	Q_Reserved6	BOOL	Reserved
8.7	Q_Reserved7	BOOL	Reserved
9.0	Q_Start	BOOL	Start query
10	Q_Error	BYTE	Internal error in query
11	Q_Reserved8	BYTE	Reserved
+32	Component		
0	C_AddressMode	BYTE	Addressing mode of the module
1	C_Reserved1	BYTE	Reserved
2	C_ComponentID	WORD	Hardware ID of the component (internal)
+36	Error		
0	E_ErrorNo	WORD	Index of the requested/actual error
2.0	E_LastError	BOOL	Is set when E_ErrorNo does not equal 0. Value TRUE if E_ErrorNo is the index of the last error otherwise FALSE
2.1	E_Reserved	ARRAY [1 to 15] BOOL	Reserved
+40	State		
0	S_Hierarchy	BYTE	Reserved
1	S_Periphery	BYTE	Reserved
2.0	S_SupFault	BOOL	The component cannot be reached
2.1	S_NotAvailable	BOOL	The component does not exist
2.2	S_Faulty	BOOL	The component is faulty, the "Alarm" section is not empty
2.3	S_MoreErrors	BOOL	There are more errors than system diagnostics can store
2.4	S_Maintenance1	BOOL	A maintenance request is pending
2.5	S_Maintenance2	BOOL	A maintenance demand is pending
2.6	S_Deactivated	BOOL	The component was deactivated *)
2.7	S_Reserved2	BOOL	Reserved
3.0	S_SubFault	BOOL	A subcomponent is faulty
3.1	S_SubMaintenance1	BOOL	A maintenance request is pending for a subcomponent
3.2	S_SubMaintenance2	BOOL	A maintenance demand is pending for a subcomponent
3.3	S_SubDeactivated	BOOL	A subcomponent is deactivated
3.4	S_Reserved4	BOOL	Reserved
3.5	S_Reserved5	BOOL	Reserved
3.6	S_Reserved6	BOOL	Reserved

Address	Name	Data type	Description
3.7	S_Reserved7	BOOL	Reserved
4	S_TIAMS	DWORD	Maintenance state of the component
8	S_TIAMSChannelExist	DWORD	Maintenance state: Configured channels
12	S_TIAMSChannelOK	DWORD	Maintenance state: Faulty channels
16	S_ChannelCount	WORD	Number of channels; valid only when Q_WithSubComponent is set
18.0	S_ChannelVector	ARRAY [0 to 255] BOOL	Number of channels affected; valid only when Q_WithSubComponent is set
+90	Alarm		
0	A_ComponentID	WORD	Hardware ID of the component (internal)
2	A_TextID1	WORD	ID of the first error text
4	A_TextLexikonID1	WORD	ID of the first error text lexicon
6	A_HelpTextLexikonID1	WORD	ID of the first help text lexicon
8	A_MapTextID	WORD	ID of the first error text (HMI)
10	A_MapHelpTextID	WORD	ID of the first help text (HMI)
12	A_TextID2	WORD	ID of the second error text
14	A_TextLexikonID2	WORD	ID of the second error text lexicon
16	A_HelpTextLexikonID2	WORD	ID of the second help text lexicon
18	A_MapTextID2	WORD	ID of the second error text (HMI)
20	A_MapHelpTextID2	WORD	ID of the second help text (HMI)
22	A_AlarmID	DWORD	Alarm number
26	A_ValueCount	WORD	Number of further bytes occupied (12)
28	A_AssociatedValue	ARRAY [1 to n] WORD	Associated values of the alarm n = A_ValueCount / 2 (= 6)
+130	SubComponent		
0	U_SubComponentCount	WORD	Number of subcomponents
2	U_SubComponentFault	ARRAY [1 to n] BYTE	List of subcomponents "n" depends on the configuration **)

*) If the component was deactivated, the index of the requested/actual error is not changed and "E_LastError" is set to "true". The tag area of the alarm is also not filled in.

**) The list of subcomponents is valid only when Q_WithSubComponent is set. The ARRAY has a status byte for each configured component. For a master, the ARRAY contains the status of the configured stations sorted in ascending order according to the station ID. For a station, the ARRAY contains the status of the configured slots sorted in ascending order according to slot number. This field can contain a maximum of 4096 entries (for an IO system): only the actual maximum size is displayed.
The status byte for each lower-level component is defined as follows:
Bit 0 = SubFault: the component cannot be reached
Bit 1 = Fault: the component is not available or faulty
Bit 2 = Maintenance1: the component has reported a need for maintenance
Bit 3 = Maintenance2: the component has reported a need for maintenance
Bit 4 = Deactivated: the component was deactivated
Bit 5 = SubFault: a subcomponent is faulty
Bit 6 = SubMaintenance1: the component has reported a need for maintenance
Bit 7 = SubMaintenance2: a subcomponent has reported a need for maintenance

8.1.6.9 PROFINET IO DB

Interface for the PROFINET IO DB

The generated data block represents the current status of all configured IO devices and can describe the status of a device in greater detail in response to a query from an HMI device. The data block is generated dynamically and depends on the hardware configuration. The DB uses the diagnostics FB generated by system diagnostics to access the diagnostics data. The current status of the devices is entered in the DB directly by this FB.

Only one HMI device (for example OP, MP, PC) can access the DB. If several HMI devices are connected, simultaneous access is interlocked using the HMI_ID tag.

Note

The diagnostics downstream from an IE/PB-Link is restricted.

Structure of the PROFINET IO DB

Address	Name	Data type	Description
0	HMI_ID	WORD	No. of the OP that uses the DB (0 = unused)
2	System_No	WORD	No. of the IO system to be investigated
4	Device_No	WORD	No. of the IO device to be investigated
6.0	Enable	BOOL	Calls up the errors of the specified device
6.1	Next_Error	BOOL	Calls up the next error of the specified device
6.2	Busy	BOOL	Busy = 1; evaluation active
6.3	More_Errors	BOOL	There are more error messages
7	Device_Status	BYTE	Status of the affected device

Address	Name	Data type	Description
8	Offset_System_Header	WORD	Address of Detail_IO_Sys[n] in the system being evaluated
10	Offset_System_Array	WORD	Address of IO_Sys[n] in the system being evaluated
12	Vendor_ID	WORD	Vendor ID is filled out if it is supported by the CPU
14	Device_ID	WORD	Device ID is filled out if it is supported by the CPU
16	Error_Level	BYTE	Error level 1=IO device, 2=module, 3=submodule, 4=channel
17		BYTE	Reserved
18	Module_No	WORD	No. of the module affected
20	Submodule_No	WORD	No. of the submodule affected
22	Channel_No	WORD	No. of the channel affected
24	Error_Cat	DWORD	Category of the error (lexicon ID)
28	Help_Cat	DWORD	Category of the error in the help lexicon
32	Error_No	DWORD	Number of the error (index in help lexicon)
36	Map_ErrorNo	WORD	The number of the error text
38	Map_HelpNo	WORD	The number of the help text
40	Number_IO_Sys	WORD	Number of configured IO systems
42	Systems_Status	WORD	Overview of all IO systems

Dynamic tag area

	Name	Type	Comment
Once	Detail_IO_Sys	Struct[n]	Array of structures per IO system
Per IO system	System_No	Word	System number
	Max_Num_Dev	Word	Maximum ID of the configured devices
	Offset	Word	Offset to the start of the field in bytes relative to Detail_IO_Sys
	Devices_Affected	Word	Number of devices affected
	Offset_Status	Word	Offset to the start of the IO_Sys_Status field in bytes regardless of Detail_IO_Sys
Per device	IO_Sys_<n>	ARRAY OF WORD[n]	Status of the groups, 1 bit for 16 devices. The table is large enough to include all configured devices (Max_Num_Dev).

Status of a device in the overview IO_Sys_<n>

Status	OK	faulty	failed	not configured
Coding (bit b+1, bit b)	00	01	10	11

Byte	N				N+1			
Bit	6-7	4-5	2-3	0-1	6-7	4-5	2-3	0-1
IO_Sys_<n>[0]: Device number	4	3	2	1	8	7	6	5
IO_Sys_<n>[1]: Device number	12	11	10	9	16	15	14	13
...								

Status of a device group in the overview IO_Sys_Status_<n>

Status	All devices of the group are OK or not configured	At least one device of the group is faulty or failed
Coding	0	1

Byte	N			N+1		
Bit	7	...	0	7	...	0
IO_Sys_Status_<n>[0]: Device number	113-128	17-112	1 - 16	241 - 256	145 - 240	129 - 144
Group	8	2 - 7	1	16	10 - 15	9
...						

8.1.6.10 PROFIBUS DP DB

Interface for the PROFIBUS DP DB

The generated data block represents the current status of all configured stations of the current DP master system and can describe the status of all DP slaves in greater detail in response to a query. The data block is generated dynamically and depends on the hardware configuration. The DB uses the diagnostics FB generated by system diagnostics to access the diagnostics data. The current status of the DP slaves is entered in the PROFIBUS DB directly by this FB.

During the processing of the PROFIBUS DB, all interrupts are delayed.

Note

Diagnostics is not possible with the master system of an IE/PB-Link. Diagnostics is performed using the PROFINET IO DB.

"Manual" mode

In this mode, all errors of the selected station are displayed one after the other.

"Automatic" mode

In this mode, the errors of all configured stations are displayed one after the other.

Structure of the PROFIBUS DP DB

Address	Name	Data type	Description
0	DP_MASTERSYSTEM	INT	Number of the DP master system
2.0	EXTERNAL_DP_INTERFACE	BOOL	External DP interface (CP/IM)
2.1	MANUAL_MODE	BOOL	Mode
2.2	SINGLE_STEP_SLAVE	BOOL	Move on to the next station affected
2.3	SINGLE_STEP_ERROR	BOOL	Move on to the next error
2.4	RESET	BOOL	DP master system (number and interface) are adopted: everything re-initialized
2.5	SINGLE_DIAG	BOOL	DP slave single diagnostics
3	SINGLE_DIAG_ADR	BYTE	DP slave address in single diagnostics
4.0	ALL_DP_SLAVES_OK	BOOL	Group indicator showing whether or not all DP slaves are operating error-free
5	SUM_SLAVES_DIAG	BYTE	Number of stations affected (faulty or failed)
6	SLAVE_ADR	BYTE	Station number of the current station

Address	Name	Data type	Description
7	SLAVE_STATE	BYTE	Status of the station: 0:o.k. 1:failed 2:faulty 3:not configured/diagnostics not possible
8	SLAVE_IDENT_NO	WORD	PROFIBUS identification number
10	ERROR_NO	BYTE	Current error number
11	ERROR_TYPE	BYTE	1:Slot diagnostics (general information) 2:Module status 3:Channel diagnostics to DP standard 4:S7 diagnostics (DS0/DS1) 5:Device diagnostics (vendor-specific) 6:Cable diagnostics (diagnostics repeater) 7: Decoded device diagnostics
12	MODULE_NO	BYTE	Slot number.
13	CHANNEL_NO	BYTE	Channel number.
14	CHANNEL_TYPE	BYTE	Channel type
15	CHANNEL_ERROR_CODE	BYTE	Error code
16	CHANNEL_ERROR_INFO_1	DWORD	Channel error code 1
20	CHANNEL_ERROR_INFO_2	DWORD	Channel error code 2
24	DIAG_COUNTER	BYTE	Sum of all the diagnostics information of the displayed station
25.0	DIAG_OVERFLOW	BOOL	Diagnostics overflow
25.1	BUSY	BOOL	Evaluation active
932 - 1176	DIAG_DAT_NORM	BYTE [1 to 244]	Slave diagnostics data
1176 - 1191	CONFIG_SLAVES	DWORD [1 to 4]	Configured slaves
1192 - 1207	EXIST_SLAVES	DWORD [1 to 4]	Existing (addressable) slaves
1208 - 1223	FAILED_SLAVES	DWORD [1 to 4]	Failed slaves
1224 - 1239	FAULTY_SLAVES	DWORD [1 to 4]	Faulty slaves
1240 - 1255	AFFECT_SLAVES	DWORD [1 to 4]	Slaves affected (faulty or failed)
1256 - 1271	AFFECT_SLAVES_MEM	DWORD [1 to 4]	Stored affected slaves (internal)
1272 - 1397	DIAG_CNT	BYTE [1 to 126]	Number of diagnostics per slave
1404	ERROR_CAT	DWORD	Text lexicon ID of the error text
1408	HELP_CAT	DWORD	Text lexicon ID of the help text
1412	ERROR_NO	DWORD	Text ID in the text lexicon
1416	MAP_ERRORNO	WORD	Error ID
1418	MAP_HELPNO	WORD	Help text ID

Address	Name	Data type	Description
1420	MASTERSTATUS_FAILED	BOOL [1 to 32]	True if at least one station of the PROFIBUS master system (1 to 32) has failed
1424	MASTERSTATUS_FAULTY	BOOL [1 to 32]	True if at least one station of the PROFIBUS master system (1 to 32) is faulty

8.1.6.11 Displaying settings of system diagnostics

Requirement

You are in the "Properties" area of the Inspector window of the required PLC.

Procedure

To display and edit the settings for system diagnostics, follow these steps:

1. Click "System diagnostics" in the area navigation of the Inspector window. In the right-hand part of the Inspector window, you can display and edit the various settings.

8.1.6.12 Basic settings

General settings

Here, you can decide which options will be enabled when the diagnostic block executes.

Enabling system diagnostics for this PLC (RSE)

This option is enabled by default.

NOTICE
If you disable this option, no diagnostics data is read out in the Web server and diagnostics viewer either even if the option is enabled there. If you disable this option on a PLC on which it was previously enabled, all the diagnostic data is deleted the next time you compile the hardware configuration.

Send alarms

This option is enabled by default. The diagnostics block then sends alarms as a reaction to system errors.

Download system diagnostics block when downloading hardware configuration

Enable this option if you want the diagnostics blocks to be generated or updated each time the hardware is compiled. So make sure that the generated alarms are up to date even after modifying the hardware configuration.

Settings for configuring alarms

You can adapt the structure and texts of the alarms created by system diagnostics and the alarm attributes freely for each reporting component.

Alarming component

Select the component (for example module or rack) for which you want to configure the alarm text.

Available alarm information

Select the entries you want to include in the alarm and click of one of the arrow buttons to include the dynamic information in the event text or info text. Regardless of your selection, the start of each alarm indicates whether it is entering or exiting state or is an alarm requiring acknowledgment and when the error/fault occurred.

See also

Adapting event text/info text (Page 634)

Adapting event text/info text

You can adapt texts to match your requirements by positioning the cursor on the required position in the box and modifying the text. Event texts and info texts can have one or more lines. Tags output by the software (for example <rack number>) are displayed in pointed brackets. If you modify these, they are no longer tags.

See also

Settings for configuring alarms (Page 634)

Attributes

The default alarm attributes "Priority", "Group ID" and "Display class" and the options "Logging" and "With acknowledgement" can be set separately for each individual reporting component.

You will find more detailed information on the alarm attributes in the description of "Configuring alarms".

Settings for system diagnostics blocks

As default, the following values are set for the system diagnostics blocks:

Block	Name	Number
Diagnostics FB	RSE_FB	49
Diagnostics DB	RSE_DB	49
first global DB	RSE_GLOBAL_DB	50
Diagnostics FC	RSE_FC	49

You can, however, assign different numbers and/or names for these blocks as long as they are not already being used.

Settings for diagnostics support

With the help of diagnostics support, diagnostics data is made available in special status DBs by the diagnostics FB. This can then be displayed graphically, for example on an HMI device.

Status DBs

Enable the required status DB to read out the current system status of configured components via this data block.

The following status DBs are set as defaults:

Block	Name	Number
Diagnostics status DB	RSE_DIAGNOSTIC_STATUS_DB	127
PROFINET IO DB	RSE_PROFINET_IO_DB	126
PROFIBUS DP DB	RSE_PROFIBUS_DP_DB	125

You can, however, assign different names and/or number for these blocks as long as they are not already being used.

If the "Enable Web server on this module" functionality is enabled in the properties of the CPU, the diagnostics status DB must be enabled. As default, DB 127 is generated if the CPU supports the Web server and if this is enabled in the properties of the CPU.

Note

The status DBs "PROFINET IO DB" and "PROFIBUS DP DB" only have limited diagnostics functionality. It is therefore preferable to use the "diagnostics status DB".

Extended diagnostics settings

This query is only possible if the PLC includes the "D_ACT_DP" instruction.

Enable the option "Query for status "activated/deactivated" after PLC startup" if you want the status of slaves to be queried.

Enable the "Send alarm if status changes from/to activated or deactivated" if you want an alarm to be output when there is a status change. To use this option, execute the "D_ACT_DP" instruction in mode 3/4 and select the diagnostics DB call in OB 86 in the "OB configuration" group.

8.1.6.13 Advanced settings

Settings for OB configuration

Here, you can make the settings you require for OBs.

- Error OBs
- Cyclic and startup OBs

The OBs that are supported by system diagnostics and the CPU are displayed in the "OB" column.

The "Exists" column indicates whether the OB already exists. This column cannot be edited.

In the "Create OB" column, you can choose whether or not the OB should be generated if it does not already exist.

In the "Call system diagnostics block" column, you can set the system diagnostics FB call if the OB already exists or will be generated. You can also remove an existing call again. If OBs exist that were not written in LAD/FBD or STL, it is not possible to insert or delete the call.

Note

- Error OBs: Before the function will execute correctly, system diagnostics must be called in all supported error OBs. There is an automatic check to determine whether or not these requirements are met.
- Cyclic and startup OBs: Before the function will execute correctly, system diagnostics must either be called in OB 1 or in a cyclic interrupt OB (30 - 38) and in all startup OBs. There is an automatic check to determine whether or not these requirements are met.

See also

Supported error OBs (Page 623)

Settings for "PLC in STOP"

Here, you can decide whether or not the PLC changes to STOP mode if a system error within an error class occurs.

The following table shows the available error classes and their meaning:

Error class	Meaning
Rack error	Errors triggered by OB 86.
Component error	• Errors in data record 0 triggered by OB 82 (global module errors) • Remove/insert module error triggered by OB 82 (DPV0) or OB 83 (central module, DPV1 and PROFINET)
Channel error	Channel errors triggered by OB 82 (data record 1, standard DP slave channel error, PROFINET channel error)
Subcomponent error	Error on a subcomponent with a subslot >= 1.

Note

In "Startup" mode, the PLC does not change to STOP regardless of the settings you made.

See also

Assignment of error classes to system errors (Page 639)

Assignment of error classes to system errors

The following table shows which system error is assigned to which error class:

Hardware	Error	Error class
Central		
Rack	Failure	Rack error
Power supply module/CPU	Power supply error	- *
H-CPU	Loss of redundancy	- *
	Return of redundancy	- *
Module	Removal/insertion of module or wrong module type	Component error
	Data record 0	Component error
	Channel error	Channel error
DP master	Failure	Rack error
IO controller	Failure	Rack error
AS-i master	Failure	Rack error
PROFIBUS DP		
DP station	Failure	Rack error
	Vendor-specific diagnostics	- *
Head	Vendor-specific diagnostics	- *
Module	Removal/insertion of module or wrong module type	Component error
	Data record 0	Component error
	Channel error	Channel error
Diag. rep head	Specific errors of the diagnostic repeater	- *
Head ET 200 B, C, U, Eco	Disrupted	Component error
H-station	Failure	Rack error
Head H-station	Loss of redundancy	- *
PROFINET IO		
IO device	Failure	Rack error
IO device head module	Vendor-specific errors	- *
	Channel error	Channel error
	Maintenance	- *
	Data record 0	Component error
	Channel error for the entire head module (subslot = 0)	Component error
IO device head submodule (PDEV)	Channel error	Channel error
	Maintenance	- *
	Data record 0	Component error
	Channel error for the entire head submodule (subslot = 0)	Subcomponent error

Hardware	Error	Error class
Module	Removal/insertion of module or wrong module type	Component error
	Data record 0	Component error
	Channel error (channel 0...7FFF)	Channel error
	Channel error for the entire module (subslot = 0)	Component error
	Maintenance (channel 0...7FFF)	- *
	Maintenance (entire module)	- *
Submodule	Removal/insertion of module or wrong module type	Subcomponent error
	Data record 0	Component error
	Channel error (channel 0...7FFF)	Channel error
	Channel error for the entire submodule (subslot >= 1)	Subcomponent error
	Maintenance (channel 0...7FFF)	- *
	Maintenance (entire submodule)	- *
IE/PB-Link	Failure	Rack error
PROFIBUS station downstream from a link	Failure	Rack error
AS-i slave		
AS-i slave PROFIBUS/central channel error	Failure	- *
AS-i slave PROFINET module	Failure	- *

* The CPU does not change to STOP mode.

Note

In "startup" mode, the CPU does not change to STOP.

See also

Settings for "PLC in STOP" (Page 638)

8.1.7 Displaying alarms

8.1.7.1 Overview of the alarm display

The "Alarm display" function can be used to output asynchronous alarms of diagnostics events and user-defined diagnostics alarms as well a alarms from ALARM instructions.

From the alarm display, you can also start the alarm editor with the "Edit alarm" shortcut menu command and then create user diagnostic alarms.

Icons

The following table shows the icons and their functions:

Icon	Function
Archive view	Shows the alarms located in the archive.
Active alarms	Shows the currently active (pending) alarms. Alarms that must be acknowledged are shown in bold characters.
Ignore	Ignores the arrival of alarms, These alarms are neither shown in the window nor stored in the archive.
Acknowledge	Confirms the selected alarm as read. Alarms requiring acknowledgment are shown in bold characters.
Clear archive	Deletes all alarms in the archive.
Export archive	Exports the current alarm archive to a file in xml format.
123 HEX Decimal/Hexadecimal	Displays the alarm numbers in decimal/hexadecimal notation.

8.1.7.2 Archive view

In the archive view, alarms are displayed and archived according to the time they appear. You can set the size of the archive (between 200 and 3000 alarms) with the menu command "Options > Settings > Online & Diagnostics". If the selected archive size is exceeded, the oldest alarm it contains is deleted.

Alarms that must be acknowledged are displayed in bold characters and can be acknowledged with the context menu command "Acknowledge alarm(s)".

The archive is constantly updated and does not need to be saved explicitly.

8.1.7.3 Layout of the alarms in the archive view

In the archive view, all events occurring on the selected CPUs are logged. A new entry is created for each individual event and shown as a further row in the table.

Table structure

All attributes of the alarms can be shown as columns. You can show or hide individual columns as well as modify the width and order of the columns. These settings are saved when the project is closed.

The alarms can be displayed in one or more rows. In the single row display, only the first row of the multiple-row alarm data is displayed.

The alarms either require acknowledgement or do not require acknowledgment. The alarms requiring acknowledgment that have not yet been acknowledged are highlighted in bold print and can be acknowledged either context-sensitive with the button in the toolbar or with the shortcut menu command "Acknowledge alarm(s)".

8.1.7.4 Receiving alarms

To allow alarms to be displayed, you must first set the receipt of alarms for each CPU.

Procedure

Toe receive alarms, follow these steps:

1. Double-click on the "Online & Diagnostics" folder of the relevant CPU in project navigation.
2. Click the "Settings" group in area navigation.
3. Select the option "Receive alarms".

Or:

1. Select the "Online & Diagnostics" folder of the relevant CPU in project navigation.
2. Open the "Online" menu and select "Receive alarms".

Or:

1. Select the relevant CPU in the device or network view.
2. Select the "Receive alarms" command in the context menu.

Or:

1. Select the CPU in project navigation.
2. Select the "Receive alarms" command in the context menu.

8.1.7.5 Export archive

To archive alarms, you can export the archive. Follow these steps:

1. Go to the archive view.
2. Click the "Export archive" button.
3. In the dialog that opens, select the path to export the archive.

Result

The archive is saved as an xml file at the location you selected.

8.1.7.6 Clear archive

The archive is organized as a ring buffer, in other words, when it is full, the oldest alarms are deleted from the archive. With the "Clear archive" button, you can delete the entire archive.

Procedure

To clear the archive, follow these steps:

1. Click the "Clear archive" button in the toolbar of the alarm display.

8.1.7.7 "Active alarms" view

The "Active alarms" view is an image of the alarm acknowledgement memory of the selected CPU(s).

8.1.7.8 Layout of the alarms in the "Active alarms" view

The "Active alarms" view represents an image of the alarm acknowledgement memory of the selected CPUs. One entry is shown in the table per active alarm. Events of an alarm ("incoming", "outgoing" and "acknowledged") are displayed in one row.

Table structure

All attributes of the alarms can be shown as columns. You can show or hide individual columns as well as modify the width and order of the columns. These settings are saved when the project is closed.

The alarms can be displayed in one or more rows. In the single row display, only the first row of the multiple-row alarm data is displayed.

The alarms either require acknowledgement or do not require acknowledgment. The alarms requiring acknowledgment that have not yet been acknowledged are highlighted in bold print and can be acknowledged either context-sensitive with the button in the toolbar or with the shortcut menu command "Acknowledge alarm(s)".

8.1.7.9 Status of the alarms

Depending on whether you are in the "Active alarms" view or the archive view, the displayed alarms may have a different status.

Status of the alarms in the "Active alarms" view

- I: Alarm came
- IA: Alarm came and was acknowledged
- IO: Alarm has gone

If more signal changes occur than can be sent (signal overflow), OV is displayed as the status and the status is shown in red.

Status of the alarms in the archive view

- No information: only with alarms generated by the PG/PC and displayed in the "Archive" tab, for example logon status, connection abort, mode changes
- I: Alarm came
- A: Alarm came and was acknowledged
- O Alarm has gone
- D: The alarm was deleted.

If more signal changes occur than can be sent (signal overflow), OV is displayed as the status and the status is shown in red.

8.1.7.10 Acknowledging alarms

Alarms that must be acknowledged are shown in bold characters.

Procedure

To acknowledge an alarm, follow these steps:

1. Select the required alarm or alarms from the table.
2. Click the "Acknowledge" button.

Note

You can select more than one alarm to acknowledge at the same time. To do this, hold down the <Ctrl> key and then select the alarms you want to acknowledge.

Result

The selected alarm was acknowledged and is then shown in normal characters.

Note

In the "Active alarms" view, acknowledged alarms that have already gone are no longer displayed.

8.1.7.11 Ignoring alarms

Ignoring alarms

To ignore alarms, follow these steps:

1. Click the "Ignore" button.

   The icon is shown on a gray background.

Result

From this point onwards, all alarms will be ignored. A message is created in the archive view indicating that the display of alarms and events is disabled.

Canceling the ignoring of alarms

To cancel the ignoring of alarms, follow these steps:

1. Click the "Ignore" button.

   The icon is shown on a white background.

Result

All alarms, in other words, even the alarms currently pending on the CPU while the "Ignore alarms" function was active, are displayed again from this point onwards. A message is created in the archive view indicating that the display of alarms and events is enabled again.

8.1.7.12 Keyboard commands in the alarm display

Alarm display

Function	Shortcut keys
Select all alarms	Ctrl+A
Acknowledge all selected alarms	Ctrl+Q

8.2 Device and network diagnostics

8.2.1 Hardware diagnostics

8.2.1.1 Overview of hardware diagnostics

Principal methods of hardware diagnostics

Principal methods of hardware diagnostics

Hardware diagnostics can be performed as follows:

- Using the Online and Diagnostics view
- Using the "Online Tools" task card
- Using the "Diagnostics > Device Info" area of the Inspector window
- Using diagnostics icons, for example, in the device view and the project tree

Structure of the Online and Diagnostics view

The Online and Diagnostics view consists of two windows alongside each other:

- The left window shows a tree structure with folders and - when you open the folder - groups.
- The right window contains detailed information on the selected folder or selected group.

The "Online access" group and the "Diagnostics" and "Functions" folders are located here:

- "Online access" group: Displays whether or not there is currently an online connection with the associated target. In addition, you can establish or disconnect the online connection.
- "Diagnostics": Contains several diagnostics groups for the selected module.
- "Functions": Contains several groups, in which you can make settings for the selected module or issue commands to the module.

Function and structure of the "Online Tools" task card

For modules with their own operating mode (such as CPUs), the "Online tools" task card allows you to read current diagnostics information and commands to the module.

If you selected a module without its own operating mode or if you selected several modules before activation of the "Online Tools" task card, the task card relates to the relevant CPU.

The "Online Tools" task card consists of the following panes:

- CPU control panel
- Cycle time
- Memory

Note

A pane is filled with content only if the module controls the associated functions and an online connection exists.

If there is no online connection to the respective module, the display "No online connection" appears in blue. If an existing online connection was disconnected, then "This target is not available" will be displayed.

Structure or the "Diagnostics" tab of the Inspector window

The "Diagnostics" tab of the Inspector window itself consists of several tabs: Of these tabs, the following is relevant for the hardware diagnostics.

- Device information

  This tab relates to all CPUs of the project to which the online connection is established. Alarms are reported here if one or more CPUs are defective or are not in RUN mode.

Determination of which of the devices that are connected online are defective

Overview of the defective devices

In the "Diagnostics > Device Info" area of the Inspector window you will obtain an overview of the defective devices that are or were connected online.

The "Diagnostics> Device Info" area of the Inspector window consists of the following elements:

- Header line with the number of defective devices
- Table with detailed information on each defective device

If you originate the establishment of an online connection to a device which is not reachable or reports one or more faults or is not in RUN mode, it will rank as defective.

Structure of the table with detailed information on the defective devices

The table consists of the following columns:

- Online status: Contains the online status as a diagnostic symbol and in words
- Operating mode: Contains the operating mode as a symbol and in words
- Device / module: Name of the affected device or the affected module
- Message: explains the entry of the previous column
- Details: The link opens the online and diagnostics view for the device, and places it in the foreground. If an online connection does not exist any longer, the link will open the connection establishment dialog.
- Help: The link supplies further information on the defect that has occurred.

See also

Determination of online status and display using symbols (Page 649)

Determination of online status and display using symbols

Determining diagnostics status online and displaying using icons

When the online connection to a device is established, the diagnostics status of the device and, if applicable, its subordinate components will be determined. The operating mode of the device is also determined, where applicable.

The following is a description of the symbols that are displayed in a specific view.

- Device view
  - The associated diagnostics icon is displayed for every hardware component (except the signal board on the CPU).
  - For hardware components with subordinate components, an "Error in subordinate component" diagnostics icon will also be displayed in the hardware component diagnostics icon if an error is present in at least one subordinate component.
  - For hardware components with their own operating mode, the operating mode icon will also be displayed to the left of or above the diagnostics icon.
- Device overview
  - The associated diagnostics icon is displayed for every hardware component.
  - For hardware components with subordinate components, an "Error in subordinate component" diagnostics icon will also be displayed in the bottom right corner of the hardware component diagnostics icon if an error is present in at least one subordinate component.

- Network view
  - The associated diagnostics icon is displayed for every device.
  - If an error is present in at least one subordinate component, then the "Error in subordinate component" diagnostics icon will also be displayed in the bottom right corner of the diagnostics icon.
- Network overview
  - The associated diagnostics icon is displayed for every hardware component.
  - For hardware components with subordinate components, an "Error in subordinate component" diagnostics icon will also be displayed in the bottom right corner of the hardware component diagnostics icon if an error is present in at least one subordinate component.
- Topology view
  - The associated diagnostics icon is displayed for every device.
  - If an error is present in at least one subordinate component, then the "Error in subordinate component" diagnostics icon will also be displayed in the bottom right corner of the diagnostics icon.
  - The associated diagnostics icon is displayed for every port.
  - Each cable between two online ports is assigned the color associated with its diagnostics status.
- Topological overview
  - The associated diagnostics icon is displayed for every hardware component.
  - For hardware components with subordinate components, an "Error in subordinate component" diagnostics icon will also be displayed in the bottom right corner of the hardware component diagnostics icon if an error is present in at least one subordinate component.
- Project tree
  - The associated diagnostics icon is displayed behind every hardware component.
  - For hardware components with subordinate components (e.g., distributed I/O, Slave_1), an "Error in subordinate component" diagnostics icon will also be displayed in the bottom right corner of the diagnostics icon if an error is present in at least one subordinate component.
  - For hardware components with their own operating mode, the operating mode icon is also displayed in the top right corner of the diagnostics icon.
  - If forcing is active on a CPU, then a red F will be displayed at the left margin of the diagnostics icon.

Diagnostics icons for modules and devices

The following table shows the available icons and their respective meaning.

Icon	Meaning
🖧	The connection with a CPU is currently being established.
🖧☒	The CPU is not reachable at the set address.
🖧	The configured CPU and the CPU actually present are of incompatible types.
🔒🖧	On establishment of the online connection to a protected CPU, the password dialog was terminated without specification of the correct password.
✓	No fault
🔧	Maintenance required
🔧	Maintenance demanded
🔧	Error
▯	The module or device is deactivated.
▯☒	The module or the device cannot be reached from the CPU (valid for modules and devices below a CPU).
▯!	Diagnostics data are not available because the current online configuration data differ from the offline configuration data.
▯	The configured module or device and the module or device actually present are incompatible (valid for modules or devices under a CPU).
▯?	The configured module does not support display of the diagnostics status (valid for modules under a CPU).
?	The connection is established, but the state of the module has not yet been determined.
⊘	The configured module does not support display of the diagnostics status.
!	Error in subordinate component: An error is present in at least one subordinate hardware component.

Note

Some modules, for example the FM 450-1, are only indicated as having a problem in the case of an error if you have enabled the diagnostics interrupt when setting the module properties.

Icons for the comparison status

The diagnostics icons can be combined at the bottom right with additional smaller icons that indicate the result of the online/offline comparison. The following table shows the available comparison icons and their meaning.

Icon	Meaning
❗	Folder contains objects whose online and offline versions differ (only in the project tree)
●	Online and offline versions of the object are different
◑	Object only exists online
◐	Object only exists offline
🟢	Online and offline versions of the object are the same

Note

If both a comparison icon and the "Error in subordinate component" diagnostics icon are to be displayed at the bottom right in the device view, the following rule applies: The diagnostics icon for the subordinate hardware component has a higher priority than the comparison icon. This means that a comparison icon is only displayed if the subordinate hardware components have no faults.

Combined diagnostics and comparison icons

The following table shows examples of icons that are displayed in the diagnostics icon.

Icon	Meaning
●	Folder contains objects whose online and offline versions differ (only in the project tree)
◑	Object only exists online

Operating mode icons for CPUs and CPs

The following table shows the available icons and their respective operating states.

Icon	Operating mode
🟩	RUN
🟨	STOP
🟨🟩	STARTUP
⏸	HOLD
🟥 ☒	DEFECTIVE
?	Unknown operating mode
⊘	The configured module does not support display of the operating mode.

Note

If forcing is active on a CPU, a red F will be displayed on a pink background at the bottom right of the operating mode icon.

Color marking of ports and Ethernet cables

The following table shows the available colors and their respective meaning.

Color	Meaning
🟩	No fault or maintenance required
🟨	Maintenance demanded
🟥	Communication error

Start online and diagnostics view

Overview of possible ways of starting the Online and Diagnostics view

You can start the Online and Diagnostics view of a module to be diagnosed at the following locations:

- Overview
- Project tree
- Device view
- Device overview
- Network view
- Network overview
- Topology view

In the following, examples are used to show how to proceed.

Requirement

The project with the module to be diagnosed is open.

Note

This requirement does not apply if you call the online and diagnostics view from the project tree after you have identified the accessible devices.

Procedure

To start the online and diagnostics view of a module, follow these steps:

1. In the project tree, open the respective device folder.
2. Double click on "Online & Diagnostics".

Or:

1. In the project tree, select the respective device folder.
2. Select the "Online & Diagnostics" command in the shortcut menu or the "Online" main menu.

Or:

1. In the project tree, open the "Online access" folder.
2. Open the folder for the interface with which you want to establish the online connection.
3. Double click on "Show/Update accessible devices".
4. Select the module to be diagnosed.
5. Select the "Online & Diagnostics" command in the shortcut menu or the "Online" main menu.

Or:

1. In the project tree, open the "Local modules" folder.
2. Select the respective device or the module that is to be diagnosed.
3. Select the "Online & Diagnostics" command in the shortcut menu or the main menu.

Or:

1. Open the device view in the device configuration.
2. Select the module to be diagnosed.
3. Select the "Online & Diagnostics" command in the shortcut menu or the "Online" main menu.

Or:

1. Open the device view in the device configuration.
2. Establish an online connection to the module to be diagnosed.
3. Double-click on the diagnostics icon above the module.

Or:

1. Open the network view in the device configuration.
2. Select the station with the module to be diagnosed.
3. Select the "Online & Diagnostics" command in the shortcut menu or the "Online" main menu.

Or:

Result

The online and diagnostics view of the module to be diagnosed will be started. If an online connection to the associated CPU had previously been created, the header bar of the Online and Diagnostics view will now have an orange background.

Note

If no online connection exists when the online and diagnostics view is started, no online information is displayed and the display fields remain empty.

Activation of the "Online Tools" task card

Activation of the "Online Tools" task card

You can activate this task card as follows:

1. Start the online and diagnostics view.
2. Click on the "Online Tools" task card.

Or:

1. Start the device view.
2. Click on the "Online Tools" task card.

Or:

1. Start the network view.
2. Click on the "Online Tools" task card.

8.2.1.2 Showing non-editable and current values of configurable module properties

Showing general properties and system-relevant information for a module

Where do I find the information I need?

The general properties and system-relevant information for a module can be found in the "General" group in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed.

Structure of the "General" group

The "General" group consists of the following areas:

- Module
- Module information
- Vendor information

"Module" area

This area shows the following data of the module:

- Short designation, for example, CPU 1214C DC/DC/DC
- Order no.
- Hardware
- Firmware
- Racks
- Slot

"Module information" area

This area shows the following data of the module that you configured during hardware configuration:

- Module name
- Installation date (not displayed for all modules)
- Additional information (not displayed for all modules)

"Manufacturer information" area

This area shows the following data of the module:

- Manufacturer
- Serial number
- Profile: Profile ID as hexadecimal number

  Note

  You will find the corresponding profile name in the profile ID table for PROFIBUS International (see "www.profibus.com").

- Profile details: Profile-specific type as hexadecimal number

  ---

  Note

  You will find the corresponding profile-specific type name in the profile-specific type table for PROFIBUS International (see "www.profibus.com").

Display configured cycle times

Where do I find the information I need?

The required information can be found in the following places:

- In the "Cycle time" group of the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed.
- In the "Cycle time" pane of the "Online Tools" task card

Structure of the "Cycle time" group in the "Diagnostics" folder of the Online and Diagnostics view

The "Cycle time" group consists of the following areas:

- Cycle time diagram (graphical display of the assigned and measured cycle times)
- Cycle time configured (display of the assigned cycle times as absolute values)
- Cycle times measured (display of the measured cycle times as absolute values)

Structure of the "Cycle time" pane of the "Online Tools" task card

The "Cycle time" pane displays the cycle time diagram and below it the measured cycle times as absolute values.

Assigned cycle times

The following assigned cycle times are displayed in the cycle time diagram and in the "Cycle time configured" area.

- Minimum cycle time
- Maximum cycle time

In the cycle time diagram, the minimum cycle time and the maximum cycle time correspond to the two markings on the time axis.

In the "Cycle time configured" area, the assigned cycle times are displayed as absolute values.

Show interfaces and interface properties of a module

Where do I find the information I need?

The interfaces and interface properties of a module can be found in the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed in the following group:

- PROFINET interface

"PROFINET interface" group

This group is divided into the following areas:

- "Ethernet address" with the "Network connection" and "IP Parameters" subareas
- "Ports"

"Network connection" subarea of the "Ethernet address" area

This subarea shows the following data of the module:

- MAC Address:

  MAC address of the interface.

  The MAC address consists of two parts. The first part ("Basic MAC address") identifies the manufacturer (Siemens, 3COM, ...). The second part of the MAC address differentiates between the various Ethernet devices. Each Ethernet module is assigned a unique MAC address.

"IP Parameters" subarea of the "Ethernet address" area

This subarea shows the following data for the module:

- IP address:

  Internet protocol address of the device on the bus (TCP/IP)
- Subnet mask:

  The subnet mask shows which part of the IP address determines the membership of a particular sub-network.
- Default router:

  If the subnet is connected via a router to other subnets, the IP address of the default router must be known. This is the only way a datagram can be forwarded with a non-matching subnet address.
- IP settings:

  Identifier for the path by which the device has obtained its IP settings (IP address, subnet mask, default router).

Identifier	Meaning
0	IP address is not initialized
1	By configuration (i.e., by the configuration loaded to the device from the device or network view)
2	Via the "Assign IP address" group of the Online and Diagnostics view
3	Via the DHCP server (i.e., the IP parameters are obtained by a special service from a DHCP server (Dynamic Host Configuration Protocol) and assigned for a limited time)

- IP setting time:

  Time stamp of the last change to the IP address directly through the Ethernet connection of the module

"Ports" area

This area shows the following data for the module:

- Ethernet ports

  Physical properties of the PROFINET interface

Properties of the PROFINET interface	Meaning
Port no.	Port number The short description of interface (X + interface no.) and port (P + port no.) is specified in parentheses. An "R" in the short description of a port means that it is a ring port.
Status	Displays the status of the port LINK LED. • Status "OK" means another device (such as a switch) is connected to the port and the physical connection is available. • Status "disconnected" means no other device is connected to the port. • Status "deactivated" means that access to the port is blocked.
Customize	Individual network settings of the device (automatic or manual)
Mode	Network settings for the speed and the transmission process

If you select a line in the port table, additional help information will be provided for the corresponding port.

8.2.1.3 Showing the current values of dynamic modules properties

Display measured cycle times

Where do I find the information I need?

The measured cycle times can be found at each of the following places:

- In the "Cycle time" group of the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed.
- In the "Cycle time" pane of the "Online Tools" task card

Structure of the "Cycle time" group in the "Diagnostics" folder of the Online and Diagnostics view

The "Cycle time" group consists of the following areas:

- Cycle time diagram (graphical display of the assigned and measured cycle times)
- Cycle time configured (display of the assigned cycle times as absolute values)
- Cycle times measured (display of the measured cycle times as absolute values)

Structure of the "Cycle time" pane of the "Online Tools" task card

The "Cycle time" pane displays the cycle time diagram and below it the measured cycle times as absolute values.

Graphical display of the measured cycle times

The following measured cycle times are displayed in the cycle time diagram:

- Shortest cycle time: Duration of the shortest cycle since the last transition from STOP to RUN

  This corresponds to the dashed gray arrow on the left in the diagram.
- Current / last cycle time: Duration of the last cycle

  This corresponds to the green arrow in the diagram. If the current / last cycle time exceeds the maximum cycle time, the arrow will turn red.

Note

If the duration of the last cycle comes close to the maximum cycle time, it may be possible that it will be exceeded. Depending on the CPU type, parameter assignment and your user program, the CPU can switch to STOP mode. If for instance you are monitoring the tags in your program, this will increase the cycle time.

If the cycle lasts longer than double the maximum cycle time, and you do not restart the maximum cycle time in the user program (by calling the extended RE_TRIGR) instruction, the CPU will switch to STOP mode.

- Longest cycle time: Duration of the longest cycle since the last transition from STOP to RUN.

  This corresponds to the dashed blue arrow on the right in the diagram.

A blue band extends between the two dashed lines; this band corresponds to the entire range of the measured cycle times. If a measured cycle time is greater than the maximum cycle time, the portion of the band that lies outside the assigned limits will be colored red.

Display of the measured cycle times as absolute values

The following measured times are displayed in the "Cycle times measured" area and in the "Cycle time" pane.

- Shortest cycle time since the last transition from STOP to RUN.
- Current/last cycle time:
- Longest cycle time since the last transition from STOP to RUN.

Showing the current status of the LEDs of a CPU

Where do I find the information I need?

The current status of the LEDs of a CPU can be found in the display area of the "CPU control panel" pane of the "Online tools" task card.

Display area of the "CPU control panel" pane of the "Online Tools" task card

This area contains the following displays:

- Station name and CPU type (short designation)
- RUN / STOP (corresponds to the "RUN / STOP" LED of the CPU)
- ERROR (corresponds to the "ERROR" LED on the CPU)
- MAINT (corresponds to the "MAINT" LED on the CPU)

Showing fill levels of all types of memory on a CPU

Where do I find the information I need?

The fill levels of all types of memory on a CPU can be found on the following two pages:

- In the display area of the "Memory" group in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed.
- In the "Memory" pane display area of the "Online Tools" task card

Display area of the "Memory" group in the "Diagnostics" folder of the online and diagnostics view

This area contains the current memory utilization of the respective module and details of the individual memory areas.

The memory utilization is shown both as a bar diagram and as a numerical value (percentage).

The following memory utilizations are shown:

- Load memory

  If no memory card is inserted, the internal load memory is displayed.

  If a memory card is inserted, the operating system only uses the inserted load memory as the load memory. This is displayed here.
- Work memory
- Retentive memory

Display area of the "Memory" pane of the "Online Tools" task card

This area contains the current memory utilization of the module. The available memory is shown both as a bar diagram and as a numerical value (percentage). The numerical value is rounded to an integer value.

Note

If a memory area is utilized to less than 1%, the available portion of this memory area is shown as "99%".

The following memory utilizations are shown:

- Load memory

  If no memory card is inserted, the internal load memory is displayed.

  If a memory card is inserted, the operating system only uses the inserted load memory as the load memory. This is displayed here.
- Work memory
- Retentive memory

See also

Load memory (Page 535)

Work memory (Page 535)

Retentive memory areas (Page 537)

8.2.1.4 Checking a module for defects

Determining the diagnostic status of a module

Where is the diagnostics status of a module displayed?

The diagnostic status of a module is displayed in the "Diagnostic status" group in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed.

The "Diagnostics status" group consists of the following areas:

- Status
- Standard diagnostics (for S7-300 and S7-400 only for non-CPU modules)

"Status" area

The following status information is displayed in this area:

- Status of the module as viewed by the CPU, for example:
  - Module available and OK.
  - Module defective.

    If the module experiences a fault and you have enabled the diagnostic error interrupt during configuration, the "Module defective" status is displayed.
  - Module configured, but not available.
- Detected differences between the configured and the inserted module. Provided it can be ascertained, the order number will be displayed for the set and actual type.

The scope of the displayed information depends on the selected module.

"Standard diagnostics" area

The following diagnostics information for non-CPU modules is displayed in this area:

- Internal and external faults that relate to the overall module
- Associated diagnostics events

Examples of such diagnostics information are:

- Entire backup failed
- Module defective

Note

Diagnostic interrupts

A diagnostic interrupt can be reported to the CPU only if the module has diagnostic interrupt capability and the diagnostic interrupt has been enabled.

The display of the diagnostic interrupt is a snapshot. Sporadic module defects can be identified in the diagnostics buffer of the respective CPU.

Reading out the diagnostics buffer of a CPU

Where do you read out the diagnostics buffer of a CPU?

You read out the diagnostics buffer of a CPU in the "Diagnostics buffer" group in the "Diagnostics" folder in the Online and Diagnostics view.

Structure of the "Diagnostics buffer" group

The "Diagnostics buffer" group consists of the following areas:

- "Events"
- "Settings"

Diagnostics buffer

The diagnostics buffer is used as a log file for the diagnostics events that occurred on the CPU and the modules assigned to it. These are entered in the order of their occurrence, with the latest event shown at the top.

"Events" area

The "Events" area consists of the following elements:

- Check box "CPU time stamp takes into account local PG/PC time"
- Event table
- "Freeze display" or "Cancel freeze" button
- Details of the event: Event no., event ID, description, time stamp, incoming/outgoing information
- "Help on event", "Open block", "Save as ..." buttons

Check box "CPU time stamp takes into account local PG/PC time"

If you have not activated the check box, the diagnostics buffer entries are shown with the module time.

If you have activated the check box, the diagnostics buffer entries are shown with the time given by the following formula:

Displayed time = module time + time zone offset on your programming device / PC

This requires the module time to be identical to UTC.

You should use this setting if you wish to see the times of the diagnostics buffer entries for the module expressed in the local time of your programming device / PC.

Selecting or clearing the check box immediately changes the times displayed for the diagnostics buffer entries.

Note

If you use the "WR_SYS_T" instruction in your program or if you set the real-time clock of the CPU using an HMI device instead of using UTC, we recommend that you clear the "CPU time stamp takes into account local PG/PC time" check box. In this case, the module time is the sole time of concern.

Event table

The following information is displayed in the table for each diagnostics event:

- Sequential number of the entry

  The first entry contains the latest event.
- Date and time of the diagnostics event

  If no date and time are shown, the module has no integral clock.
- Short name of the event and, if applicable, the reaction of the CPU

  ### Note

  If an individual parameter of a text cannot be determined, the character string "###" is shown in its place.

  If no display text is yet available for new modules or new events, the numbers of the events and the individual parameters are stated as hexadecimal values.
- Icon for information related to incoming/outgoing status

  The following table shows the available icons and their respective meaning.

Icon	Meaning
✉←	Incoming event
✉→	Outgoing event
i	Incoming event to which there is no independent outgoing event
✉	User-defined diagnostics event

- For S7-1200 CPUs only: Icon for the severity of the event

  The following table shows the available icons and their respective meaning.

Icon	Meaning
✓	No maintenance and/or no fault
🔧	Maintenance required
🔧	Maintenance demanded
🔧	Error

You can change the order of the columns, adjust the column widths, and remove and add individual columns in the event table. Sorting is possible, but only by the sequential numbers.

"Freeze display" or "Cancel freeze" button

The "Freeze display" or "Cancel freeze" button is only enabled when there is an online connection to the CPU.

The default setting is "Freeze display".

The following happens when you click the "Freeze display" button:

- The current display of the diagnostics buffer entries is frozen.
- The labeling of the button changes to "Cancel freeze".

If an error has occurred in your system, diagnostics events can occur very quickly in succession. This produces a high update rate on the display. Freezing the display allows you to calmly examine the situation in more detail.

If the display is frozen and you click the "Cancel freeze" button, the following happens:

- The display of the diagnostics buffer entries is updated again.
- The labeling of the button changes to "Freeze display".

Note

If you freeze the diagnostics buffer display, the CPU continues to enter events in the diagnostics buffer.

Details of the event

If you select a line in the list of events, you will obtain detailed information on the respective event:

- Sequential number of the event in the diagnostics buffer.
- Event ID
- Description of the event with event-dependent additional information. Examples of this additional information:
  - Command that caused the event
  - Operating mode switch caused by the diagnostics event
- Time stamp
- For S7-1200 CPUs only: Associated I&M data (module, rack/slot, plant designation, location designation)
- Priority of the event
- Information on whether the event is an incoming or outgoing event

"Help on event" button

If you click on this button, the selected event is explained in more detail and any remedies given.

Note

If the selected event is not a CPU event, the "Help on event" button is unavailable.

"Open block" button

The following table shows if the "Open block" button is active and which function it conceals.

When is the "Open block" button active?	What happens when you click this button?
If the diagnostics event references the relative address of a block. This is the address of the command that caused the event.	The "Open block" function opens the referenced block in the offline view at the programming instruction that caused the error. This allows you to check and, if necessary, change the source code of the block at the specified place and then download it again to the CPU.
If the diagnostics event was triggered by a module.	The "Open block" function opens the Device view of the module involved.

"Save as ..." button

If you click this button, the content of the diagnostics buffer is saved in a text file. "Diagnostics", depending on the language, with the extension ".txt" is suggested as the file name. You can however change this name.

"Settings" area

The "Settings" area consists of the following elements:

- "Display events" list
- "Apply settings as default" button
- "Output event information in hexadecimal format" check box

"Display events:" list

There is an check box in this list for every event class (default setting: all check boxes are selected). If you clear a check box, the events of that event class will no longer be displayed in the "Events" area. Reselecting the check box will display the associated events once again.

"Apply settings as default" button

If you click this button, the settings will also apply to future occasions when the "Events" tab is opened.

"Output event information in hexadecimal format" check box

If you select the check box, the event IDs in the Events list of the "Events" area will be displayed in hexadecimal format. If you clear the check box, the event information is given in text form.

See also

Basic information on the diagnostics buffer (Page 678)

8.2.1.5 Changing the properties of a module or the programming device / PC

Changing the mode of a CPU

Requirement

There is an online connection to the CPU whose mode you want to change.

Procedure

To change the mode of the CPU, follow these steps:

1. Enable the "Online tools" task card of the CPU.
2. Click the "RUN" button in the "CPU control panel" pane if you want to change the CPU to RUN mode or the "STOP" button if you want to change the CPU to STOP mode.

   Note

   The only button active is the one that can be selected in the current operating mode of the CPU.

3. Acknowledge the confirmation prompt with "OK".

Or:

1. Open the "Online" menu.
2. Choose the "Start CPU" menu command if you want to set the CPU to RUN mode and "Stop CPU" if you want to set the CPU to STOP mode.

   Note

   The only button that is active is the one that can be chosen in the current operating mode of the CPU.

3. Acknowledge the confirmation prompt with "OK".

Or:

1. Click the "Start CPU" button in the toolbar if you want to set the CPU to RUN mode and the "Stop CPU" button if you want to set the CPU to STOP mode.

   Note

   The only button that is active is the one that can be chosen in the current operating mode of the CPU.

2. Acknowledge the confirmation prompt with "OK".

Result

The CPU will be switched to the required operating mode.

Performing a memory reset

Requirement

- There is an online connection to the CPU on which the memory reset is to be performed.
- The CPU is in STOP mode.

Note

If the CPU is still in RUN mode and you start the memory reset, you can place it in STOP mode after acknowledging a confirmation prompt.

Procedure

To perform a memory reset on a CPU, follow these steps:

1. Enable the "Online Tools" task card of the CPU.
2. Click the "MRES" button in the "CPU control panel" pane.
3. Acknowledge the confirmation prompt with "OK".

Result

The CPU is switched to STOP mode, if necessary, and the memory reset is performed on the CPU.

See also

Basics of a memory reset (Page 532)

Determining and setting the time of day on a CPU

Where do I find the functions I need?

You determine and change the time of day on a CPU in the "Set time of day" group in the "Functions" folder of the Online and Diagnostics view. This requires an online connection.

Structure of the "Set time of day" group

The "Set time of day" group consists of the following areas:

- Area for reading out and setting the time of day
- Time system (This area does not exist for S7-1200 and will not be examined here.)

Structure of the area for reading out and setting the time of day

This area consists of the following parts:

- Programming device / PC time

  Here the time zone setting, the current date and the current time setting of your programming device / PC are displayed.
- Module time

  Here the date and time values currently read from the module (for example the CPU), are converted to local time and date and displayed.

  If the "Take from PG/PC" check box is selected, when you click the "Apply" button, the date and the PG/PC time converted to UTC are transferred to the module.

  If the "Take from PG/PC" check box is not selected, you can assign the date and time for the integrated clock of the module. After clicking the "Apply" button, the date and the time recalculated to UTC time are transferred to the module.

Assigning an IP address to a PROFINET IO device

Overview

All PROFINET IO devices work with the TCP/IP protocol and therefore require an IP address for operation on Industrial Ethernet. Once an IO device has been assigned an IP address, it can be accessed via this address. This step lets you download configuration data or perform diagnostics, for example.

You have two basic options for choosing an IP address:

- You assign the IP address yourself.
- You obtain the IP address from a DHCP server. (This is not possible for some IO devices.) In this case, the client ID, the MAC address, or the device name of the IO device is communicated to the DHCP server, depending on the option selected.

Both options are described below.

Requirement

- The Ethernet LAN connection must already be established.
- The Ethernet interface of your programming device or PC must be accessible.
- The IO device that is to be assigned an IP address must be in the same IP band as the programming device or PC.

Procedure for specifying the IP address yourself

To assign the IP address you have specified yourself to the IO device, proceed as follows:

1. Open the Online and Diagnostics view of the IO device.
2. Select the "Assign IP address" group from the "Functions" folder.
3. Select the "Use IP parameters" option.
4. Click the "Accessible devices" button to identify the devices that can be accessed via MAC addresses. After the search is complete, select the IO device with the MAC address known to you. The "IP address" entry field now contains the value configured for this IO device.
5. Check this value and correct it, if necessary.
6. Enter the subnet mask.
7. If a router is to be used, select the "Use router" check box and enter its IP address.
8. Click "Assign IP address".

Procedure for obtaining the IP address from from a DHCP server.

To assign an IP address specified by a DHCP server to the IO device, proceed as follows:

1. Open the Online and Diagnostics view of the IO device.
2. Select the "Assign IP address" group from the "Functions" folder.
3. Select the "Obtain IP address from a DHCP server" option.
4. If the DHCP server is to identify the target device using its client ID, choose the "Client ID" option. Enter the client ID in the entry field of the same name.

   Note

   The client ID is a string with a maximum of 63 characters. Only the following characters are permitted: a-z, A-Z, 0-9 and - (dash)

5. Alternatively: If the DHCP server is to identify the target device using its MAC address, choose the "MAC address" option. The "Accessible devices" button now becomes active, and you select the target device as described in section "Procedure for specifying the IP address yourself".

6. Alternatively: If the DHCP server is to identify the target device using its device name, choose the "Device name" option.

   Note

   If you specify that the DHCP server is to identify the target device using its device name, you must have assigned a device name to the target device beforehand.

   ---

7. Click "Assign IP address".

Result

The IP address will be assigned to the IO device.

Resetting a CPU to the factory settings

Requirement

- There is no memory card inserted in the CPU.
- There is an online connection to the CPU that you want to reset to the factory settings.
- The CPU is in STOP mode.

Note

If the CPU is still in RUN mode and you start the reset operation, you can place it in STOP mode after acknowledging a confirmation prompt.

Procedure

To reset a CPU to the factory settings, follow these steps:

1. Open the Online and Diagnostics view of the CPU.
2. Select the "Reset to factory settings" group from the "Functions" folder.

   Note

   The MAC address is not displayed for S7-1200 CPUs.

   ---

3. Select the "Retain IP address" check box if you want to retain the IP address or the "Delete IP address" if you want to delete the IP address.

   Note

   The two check boxes mentioned are only available if the module to be reset is able to choose whether to retain or delete the IP address.

   ---

4. Click the "Reset" button.
5. Acknowledge the confirmation prompt with "OK".

Result

The module is switched to STOP mode, if necessary, and the factory settings are then reset. This means:

- The work memory and the internal load memory and all operand areas are deleted.
- All parameters are reset to their defaults.
- The diagnostics buffer is cleared.
- The time of day is reset.
- The IP address is retained or deleted depending on the setting you make.

Assigning a PROFINET device name

Basic information on assigning a name to a PROFINET IO device

Device name

Before an IO device can be addressed by an IO controller, it must have a device name. This procedure was chosen for PROFINET because names are easier to handle than complex IP addresses.

Assigning a device name to a PROFINET IO device is comparable to setting the PROFIBUS address for a DP slave.

An IO device has no device name in its delivery state. For an IO controller to address an IO device, it must first be assigned a device name using the programming device or PC. It is now ready to transfer the configuration information including the IP address during startup or exchange user data in cyclic operation.

Rules for the device name

The device name is subject to the following limitations:

- Restricted to a total of 240 characters (lower case letters, numbers, dash, or dot)
- A name component within the device name, which is a character string between two dots, must not exceed 63 characters.
- No special characters such as umlauts, brackets, underscore, slash, blank space, etc. The only special character permitted is the dash.
- The device name must not begin or end with the "-" character.
- The device name must not begin with a number.
- The device name form n.n.n.n (n = 0, ... 999) is not permitted.
- The device name must not begin with the string "port-xyz" or "port-xyz-abcde" (a, b, c, d, e, x, y, z = 0, ... 9).

Where do I find the function I am seeking?

To assign a name to a PROFINET IO device, go to the "Assign name" group in the "Functions" folder of the Online and Diagnostics view for the device. The user interface for this group differs depending on how you open the Online and Diagnostics view:

- Call from project navigation (using "Update accessible devices") or via the "Online" menu
- Call from within the project context

See also

Calling the name assignment function from the project tree or via the "Online" menu (Page 676)

Calling the name assignment function from within the project context (Page 677)

Calling the name assignment function from the project tree or via the "Online" menu

Requirement

- You have opened the Online and Diagnostics view of the PROFINET IO device from the project tree or with the "Online" menu.

Procedure

1. Open the "Functions" folder and the "Assign name" group inside this folder. The "Type" field displays the module type of the PROFINET IO device.
2. Enter the required device name in the "PROFINET device name" input box.
3. Optional: Click the "Flash LED" button on the left of the graphic to run an LED flash test for the PROFINET IO device. In this way you verify that you are naming the desired IO device.

Note

The LED flash test is not supported by all PROFINET IO devices.

4. Click "Assign name".

Result

The entered name is assigned to the PROFINET IO device.

Calling the name assignment function from within the project context

Requirement

- An online connection to the PROFINET IO device is not required.
- You have opened the Online and Diagnostics view of the PROFINET IO device from within the project context.
- The PROFINET IO device can be accessed using at least one PG/PC.

Procedure

1. Open the "Functions" folder and the "Assign name" group inside this folder. The "PROFINET device name" drop-down list displays the current name in the offline project, and the "Type" box shows the module type of the PROFINET IO device.
2. Choose a different name from the drop-down list, if necessary.

Note

In steps 3 to 5, you determine the IO devices that are present in the PROFINET subnet.

3. In the "PG/PC interface for assignment" drop-down list, select the PG/PC interface you want to use to establish the online connection.
4. Optional: Use the three check boxes to make a selection from all IO devices available online.
5. Click the icon for determining the IO devices present in the PROFINET subnet. The table is then updated.
6. Select the desired IO device in the table.
7. Optional: Click the "Flash LED" button to run an LED flash test for the PROFINET IO device. In this way you verify that you are naming the desired IO device.

Note

The LED flash test is not supported by all PROFINET IO devices.

8. Click "Assign name".

Result

The selected name is assigned to the PROFINET IO device.

8.2.1.6 Diagnostics in STOP mode

Basic information on the diagnostics buffer

Function

The operating system of the CPU enters the errors detected by the CPU and the diagnostics-capable modules into the diagnostics buffer in the order in which they occurred. This includes the following events:

- Every mode change of the CPU (POWER UP, change to STOP mode, change to RUN mode)
- Every hardware and diagnostic error interrupt

The top entry contains the most recent event. The entries in the diagnostics buffer are stored permanently. They are retained even if the power supply fails and can only be deleted by resetting the CPU to factory settings.

A diagnostics buffer entry contains the following elements:

- Time stamp
- Error ID
- Additional information specific to the error ID

Advantages of the diagnostics buffer

The diagnostics buffer offers the following advantages:

- After the CPU has changed to STOP mode, you can evaluate the last events prior to the STOP so that you can locate and identify the cause of the STOP.
- You can detect and eliminate the causes of errors more quickly and thus increase the availability of the system.
- You can evaluate and optimize the dynamic system response.

Organization of the diagnostics buffer

The diagnostics buffer is a ring buffer. The maximum number of entries for the S7-1200 CPUs is 50. When the diagnostics buffer is full and a further entry needs to be made, all existing entries are shifted by one position (which means that the oldest entry is deleted) and the new entry is made at the top position that is now free (FIFO principle: first in, first out).

Evaluation of the diagnostics buffer

The contents of the diagnostics buffer can be accessed as follows:

- Using the Online and Diagnostics view

The evaluation of events occurring prior to the error event (e.g., transition to STOP mode) allows you to obtain a picture of the possible causes or to zero in more closely or specify in more detail the possible causes (depending on the error type).

Read the detailed information about the events carefully and use the "Help on event" button to obtain additional information and possible causes of individual entries.

Note

To make the best use of the time stamp information on the diagnostics buffer entries in time-critical systems, it is advisable to check and correct the date and time of day on the CPU occasionally.

Alternatively, it is possible to perform a time-of-day synchronization using an NTP time server.

See also

Resetting a CPU to the factory settings (Page 674)

Determining the cause of a STOP of a CPU (Page 679)

Determining and setting the time of day on a CPU (Page 672)

Parameterizing the clock (Page 555)

Determining the cause of a STOP of a CPU

Requirement

The CPU you want to analyze is in STOP mode.

Procedure

To find out the reason why a CPU changed to STOP, follow these steps:

1. Open the online and diagnostics view of the CPU.
2. Select the "Diagnostics buffer" group from the "Diagnostics" folder.
3. Evaluate the events occurring prior to the transition to STOP mode. Use this to obtain a picture of the possible causes or to zero in on or specify in more detail the possible causes (depending on the error type).

   Read the detailed information about the events carefully and use the "Help on event" button to obtain additional information and possible causes of individual entries.

Result

You were able to zero in on or determine in more detail the cause of the CPU STOP.

Note

If the analysis does not enable you to overcome the problem, contact Customer Support. In this case, use the "Save as" button to back up the content of the diagnostics data to a text file and submit it to Customer Support.

See also

Reading out the diagnostics buffer of a CPU (Page 665)

8.2.1.7 Online accesses in the Online and Diagnostics view

Displaying status of the online connection

Requirement

- The associated device can be accessed using at least one PG/PC interface.

Procedure

1. Open the Online and Diagnostics view for the device whose online connection status you want to display.
2. Select the "Online access" group.

Note

The "Online access" group exists only for CPUs. However, if you have opened the Online and Diagnostics view using the "Show/update accessible devices" function, it will not be displayed.

Result

The status of the online connection is displayed in the "Status" area both graphically and in text form.

Specifying a PG/PC interface, going online

Requirement

- The associated device can be accessed using at least one PG/PC interface.
- There is currently no online connection to the relevant device.

Procedure

1. Open the Online and Diagnostics view of the device to which you want to establish an online connection.
2. Choose the "Online access" group and the "Online access" area within this group.

Note

The "Online access" group exists only for CPUs. However, if you have opened the Online and Diagnostics view using the "Show/update accessible devices" function, it will not be displayed.

3. If an online connection was established previously for the device, the associated data for this online connection will be preset in the drop-down lists. In this case, you can skip directly to the last step of this operating instruction.
4. Choose the interface type in the "Type of PG/PC interface" drop-down list.

   The "PG/PC interface for online access" drop-down list then shows only the interfaces of the programming device or PC that match the selected interface type.
5. In the "PG/PC interface for online access" drop-down list, select the programming device or PC interface via which you want to establish the online connection.
6. Optional: Click the "Properties" button to change the properties of the associated CP.
7. In the "Connection to subnet" drop-down list, select the subnet via which the device is connected to the PG/PC interface. If the device is connected directly to the PG/PC interface, select the "Local" setting.
8. If the device is accessible via a gateway, select the gateway that connects the two subnets involved in the "1st gateway" drop-down list.
9. In the "Device address" entry field, enter the IP address of the device to which you want to establish an online connection.
10. Alternatively: Click the "Show accessible devices" button and choose the device from the list of accessible devices to which you want to establish an online connection.
11. Click the "Go online" button.

Result

The online connection to the desired device will be established.

Going offline

Requirement

- There is currently an online connection to the relevant device.

Procedure

1. Open the Online and Diagnostics view of the device for which you want to disconnect the online connection.
2. Choose the "Online access" group and the "Online access" area within this group.

   Note

   The "Online access" group exists only for CPUs. However, if you have opened the Online and Diagnostics view using the "Show/update accessible devices" function, it will not be displayed.

3. Click the "Go offline" button.

Result

The online connection to the desired device will be disconnected.

Performing the flash test for a device with an online connection

Requirement

- There is currently an online connection to the relevant device.

Procedure

1. Open the Online and Diagnostics view of the device for which you want to perform a flash test.
2. Choose the "Online access" group and the "Status" area within this group.

   Note

   The "Online access" group exists only for CPUs. However, if you have opened the Online and Diagnostics view using the "Show/update accessible devices" function, it will not be displayed.

3. Click the "LED Flash test" button.

Result

- On an S7-1200 CPU, the RUN/STOP, ERROR and MAINT LEDs flash.
- On an S7-300 or S7-400 CPU, the FRCE LED flashes.

Note

The flash test cannot be performed when the FORCE function is active.

8.2.1.8 Checking PROFIBUS DP subnets for faults

Basic information on the diagnostic repeater

What is the diagnostic repeater?

The diagnostic repeater is a repeater that can monitor a segment of an RS 485 PROFIBUS subnet (copper cable) during operation and signal line errors to the DP master via a diagnostics message frame.

The diagnostic repeater detects, localizes and visualizes line errors during operation at an early stage. As a result, problems in the system are identified early and production downtimes will be minimized.

Function of the diagnostic repeater

The diagnostic repeater can perform line diagnostics on the DP2 and DP3 segments because it has a measuring circuit for these segments.

The line diagnostics run in two steps:

- Step 1: Topology determination

  You start the topology determination by calling the "DP_TOPOL" instruction in your program.

  The diagnostic repeater then determines the PROFIBUS addresses and the distance of the devices and creates a topology table.

- Step 2: Error localization

  The diagnostic repeater checks the lines during operation. It determines the distance to the point of the error and the reason for the error; it then issues a diagnostics alarm with relative information on the error location.

Display of detailed information on the determined error location

You receive detailed information on the determined error location in the Online and Diagnostics view of the diagnostic repeater.

- By means of icons
- By means of a display with graphics and text

See also

Displaying the status of the segment diagnostics using icons (Page 684)

Displaying the status of the segment diagnostics using graphics and text (Page 684)

Displaying the status of the segment diagnostics using icons

Where do I find the information I need?

The icons for the status of the segment diagnostics are available:

- In the expanded "Segment diagnostics" folder in the navigation pane of the Online and Diagnostics view of the relevant diagnostic repeater.

The diagnostics icon associated with the segment will be displayed behind the segment designation. It must be noted here that line errors will be displayed for the DP2 and DP3 segments only. The DP1 and programming device segments do not display errors in the form of a diagnostics icon; rather, they signal only a few bus errors.

Diagnostics icons

The following table shows the available icons and their meaning.

Icon	Meaning
✅	Segment is error-free
🔧	Segment contains errors
⊽	Segment is deactivated

Displaying the status of the segment diagnostics using graphics and text

Where is the status of the segment diagnostics displayed with graphics and text?

The status of the segment diagnostics will be displayed using graphics and text in the "DP1", "DP2", "DP3", and "PG" groups of the "Segment diagnostics" folder in the Online and Diagnostics view of the relevant diagnostic repeater.

Structure of the "DP1", "DP2" "DP3", and "PG" groups

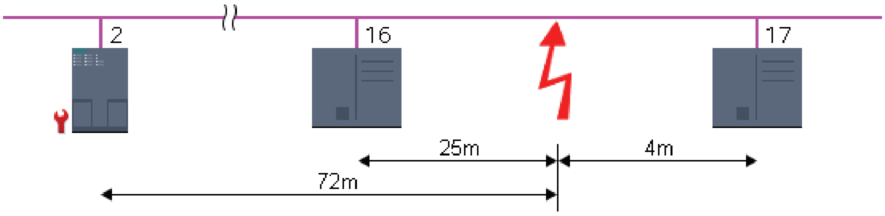
The "DP1", "DP2", "DP3", and "PG" groups consist of the following elements:

- "Error location" field
- "Error" field
- "Resolution" field
- "Help on event" button
- "Freeze display" or "Cancel freeze" button

"Error location" field

This field displays the error location graphically, provided the diagnostic repeater can determine the location.

The following picture shows an example for a line error occurring in the DP2 segment.



In this example, the diagnostic repeater has the PROFIBUS address 2, and a line error has occurred between the devices with PROFIBUS addresses 16 and 17. This line error is located 25 m from device 16, 4 m from device 17, and 72 m from the diagnostic repeater.

"Error" field

The error is explained in plain text in this field.

"Resolution" field

Here, you will find actions for resolving the error.

"Help on event" button

Click this button to obtain a more detailed explanation of the error and additional details on resolving the error, if applicable.

"Freeze display" or "Cancel freeze" button

The "Freeze display" or "Cancel freeze" button is only enabled when there is an online connection to the diagnostic repeater.

The default setting is "Freeze display".

The following happens when you click the "Freeze display" button:

- The current display of the segment diagnostics is frozen.
- The labeling of the button changes to "Cancel freeze".

If the display is frozen and you click the "Cancel freeze" button, the following happens:

- The display of the segment diagnostics is updated again.
- The labeling of the button changes to "Freeze display".

8.2.2 Connection diagnostics

8.2.2.1 Overview of connection diagnostics

Basics

Connection diagnostics, as described below, refers to the diagnostics of communication connections.

The connection diagnostics is started each time an online connection is established to a module (CPU or CP) that participates in one or more communication services. The connection status is updated automatically in the background.

In the case of one-way connections, an online connection must exist to the communication partner that has established the communication connection.

On connections configured at both ends, a distinction between the following two situations must be made:

- If there is an online connection to only one connection endpoint, only the part of the connection belonging to this connection endpoint can be diagnosed.
- If there is an online connection to both connection endpoints, both parts of the connection (and therefore the entire connection) can be diagnosed.

Basic connections diagnostics options

Connection diagnostics can be performed as follows:

- Using icons on the connection status display

  This display is generated in the connection table.
- Through detailed connection diagnostics

  This step is available in the "Diagnostics > Connection information" area of the Inspector window.

Requirement for the connection diagnostics described below

You can display the details of either all the communication connections created in the project (default) or selected communication connections in the connection table.

The connection diagnostics described in the following assume that you display the details of selected communication connections. To do this, clear the "Show all connections" option in the shortcut menu.

8.2.2.2 Displaying the connection status using icons

Content of connection table without an online connection

For a CPU or CP, the connection table lists the communication connections (including properties) configured offline, if an online connection is not established. For S7-1200, the following connections are listed:

- Configured S7 connections (for S7-1200 these are configured only with the PUT and GET instructions)
- HMI connections

Content of connection table with an online connection

After the online connection has been established, the properties of the communication connections listed offline will be expanded to include diagnostics icons for the connection status ("Online status" column).

In addition, entries for all communication connections that exist online only (e.g., connections configured using open user communication instructions, PG and OP connections) will now be added to the connection table.

For connections that exist online or offline only, the diagnostics icon at the bottom right is combined with a smaller additional comparison status icon.

Diagnostics icons for communication connections

The following table shows the diagnostics icons for communication connections.

Icon	Meaning
✅	Connection setup
🔧	Connection not setup / is being setup
❔	Connection not available

Diagnostics icons for the comparison status

The diagnostics icons for communication connections can be combined at the bottom left with additional smaller icons that indicate the result of the online/offline comparison. The following table shows the available comparison icons and their meaning.

Icon	Meaning
◑	Connection exists online only
◐	Connection exists offline only

8.2.2.3 Detailed connection diagnostics

Detailed connection diagnostics - overview

Where do I perform detailed connection diagnostics?

To perform detailed connection diagnostics, go to the "Diagnostics > Connection" information of the Inspector window.

How do I open the "Diagnostics > Connection information" area of the Inspector window?

The following options are available for opening the "Connection information" tab of the Inspector window.

- Select the line of the relevant connection in the connection table. Click the "Diagnostics" and "Connection information" tabs one after the other in the Inspector window.
- Double-click the diagnostics icon of the relevant connection in the connection table.
- This step takes you to the programming editor for a S7 communication instruction or open user communication instruction. Double-click the diagnostics icon of the instruction (3 squares back-to-back at an angle in green, yellow, red).

Structure or the "Diagnostics > Connection information" area of the Inspector window

Requirements: the content of the "Connection information" tab has been filled, and an online connection to at least one end point of the relevant connection has been established.

If a module has been selected (network view), the tab will contain the following group:

- Connection resources (for S7-1200 only)

If a connection has been selected (connection table), it will contain the following groups:

- Connection details
- Address details of the connection (for S7-1200 only)

Determining connection resources

Where do I determine connection resources?

The connection resources are obtained in the "Connection resources" group. This group is located in the "Diagnostics > Connection information" area of the Inspector window. It is displayed only if you have selected a module in the Network view.

Number of connection resources

- Maximum number: Specifies the maximum number of available connection resources of the module.
- Not assigned: Indicates how many connection resources are not yet assigned. If connection resources are already reserved for certain types of communication, then the unreserved connection resources cannot always be used for the various connection types.

Reserved and currently assigned connection resources

For the communication types indicated below, the connection resources that are reserved and currently assigned by the module will be displayed.

Communication type	Meaning
PG communication	Resources for connections between the module and programming devices (for example, for the establishment of a connection from the project tree, for online diagnostics, etc.)
HMI communication	Resources for connections between the module and HMI devices
Open user communication	Resources for connection of open user communication instructions
S7 communication	Resources for configured S7 connections, through which data can be exchanged by calling instructions in the user program.
Other communication	Specifies other assigned connection resources for which connection resources are not reserved.

Determining connection details

Where do I determine the connection details?

The connection details are obtained in the "Connection details" group. This group is located in the "Diagnostics > Connection information" area of the Inspector window.

When is the "Connection details" group filled in?

The following requirements must be met to fill in the "Connection details" group on the "Connection information" tab:

- An online connection to the end point of the relevant connection must exist.
- You have selected a line in the connection table.

Structure of the "Connection details" group

The "Connection details" group consists of the following elements:

- Local ID (hex)
- Connection type (for S7-1200 only)
- Protocol
- Connection status: icon and description
- Details
- Last status change (for S7-300 and S7-400 only)

Determining the address details of a connection

Where do I determine the address details of a connection?

The address details of a connection are obtained in the "Address details for connection" group. This group is located in the "Diagnostics > Connection information" area of the Inspector window.

For which CPUs is the "Address details for connection" group available?

The "Address details for connection" group of the "Connection information" tab is available for S7-1200 CPUs only.

When is the "Address details for connection" group filled in?

The following requirements must be met to fill in the "Address details for connection" group on the "Connection information" tab:

- An online connection to the end points of the relevant connection must exist.
- You have selected a line in the connection table.

Structure of the "Address details of connection" group

The following information is specified for the two communication partners:

- Endpoint
- Interface (not for remote partner)
- Subnetwork (not for remote partner)
- Address
- TSAP
- Protocol
- Active connection setup: Yes/no

Programming a PLC 9

9.1 Creating a user program

9.1.1 Programming basics

9.1.1.1 Operating system and user program

Operating system

Function

Every CPU comes with an integrated operating system that organizes all CPU functions and sequences not associated with a specific control task.

The tasks of the operating system, for example, include the following:

- Handling Warm restart (Page 528)
- Updating the process image of the inputs and outputs
- Calling the user program
- Detecting interrupts and calling interrupt OBs
- Detecting and handling errors
- Managing memory areas

The operating system is a component of the CPU and is already installed there upon delivery.

See also

User program (Page 694)

User program

Function

The user program contains all functions that are necessary for processing your specific automation task.

The tasks of the user program include:

- Checking the requirements for a (warm) restart using startup OBs, for example, limit switch in correct position or safety relay active.
- Processing process data, e.g. linking binary signals, reading in and evaluating analog values, defining binary signals for output, and outputting analog values
- Reaction to interrupts, for example, diagnostic error interrupt if the limit value of an analog expansion module is overshot.
- Error handling in normal program execution

You write the user program and load it into the CPU.

See also

Operating system (Page 693)

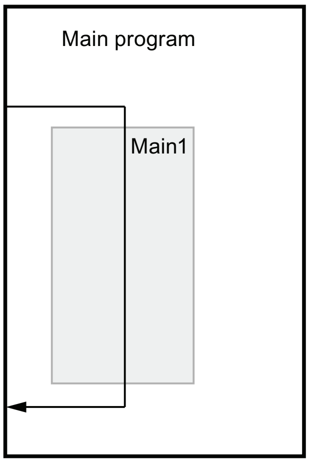
9.1.1.2 Blocks in the user program

Linear and structured programming

Linear programming

Solutions for small automation tasks can be programmed linearly in a program cycle OB. This is only recommended for simple programs.

The following figure shows a linear program schematically: The "Main1" program cycle OB contains the complete user program.



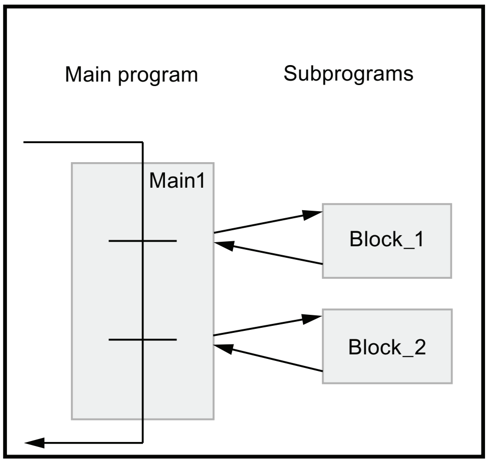
Structured programming

Complex automation tasks can be more easily handled and managed by dividing them into smaller sub-tasks that correspond to the technological functions of the process or that can be reused. These sub-tasks are represented in the user program by blocks. Each block is then an independent section of the user program.

Structuring the program offers the following advantages:

- Extensive programs are easier to program through the structure.
- Individual program sections can be standardized and used repeatedly with changing parameters.
- Program organization is simplified.
- Changes to the program can be made more easily.
- Debugging is simplified since separate sections can be tested.
- Commissioning is simplified.

The following figure shows a structured program schematically: The "Main1" program cycle OB calls subprograms one after the other that execute defined subtasks.



Overview of the block types

Block types

Different BLOCK types are available to perform tasks within an automation system. The following table shows the available block types:

Block type	Brief description
Organization blocks (Page 696) (OB)	Organization blocks define the structure of the user program.
Functions (Page 697) (FC)	Functions contain program routines for recurring tasks. They have no "memory".
Function blocks (Page 697) (FB)	Function blocks are code blocks that store their values permanently in instance data blocks, so that they remain available even after the block has been executed.
Instance data blocks (Page 699)	Instance data blocks are assigned to a function block when it is is called for the purpose of storing program data.
Global data blocks (Page 698)	Global data blocks are data areas for storing data that can be used by any blocks.

Organization blocks (OB)

Definition

Organization blocks (OBs) form the interface between the operating system and the user program. They are called by the operating system and control, for example, the following operations:

- Startup characteristics of the automation system
- Cyclic program processing
- Interrupt-driven program execution
- Error handling

You can program the organization blocks and at the same time determine the behavior of the CPU. Various organization blocks are available to you depending on the CPU used.

For more information on organization blocks, refer to the descriptions of the mode of operation of the CPUs in the References section of "Configuring Hardware and Networks".

Start information of organization blocks

At the start of many organization blocks the operating system outputs information that can be evaluated in the user program. The descriptions of the organization blocks contain information on whether information is output and on which information is output.

See also

Creating organization blocks (Page 844)

Functions (FCs)

Definition

Functions (FCs) are logic blocks without memory. After the function has been executed, the data in the temporary variables is therefore lost. Functions can use global data blocks to store data permanently.

Application

A function contains a program that is executed when the function is called by another logic block. Functions can be used, for example, for the following purposes:

- To return function values to the calling block, e.g. for mathematical functions
- To execute technological functions, e.g. individual controls using bit logic operations

A function can also be called several times at different points in a program. As a result, they simplify programming of frequently recurring functions.

See also

Creating functions and function blocks (Page 845)

Function blocks (FB)

Definition

Function blocks are logic blocks that store their input, output and in-out parameters permanently in instance data blocks, so that they remain available even after the block has been executed. Therefore they are also referred to as blocks "with memory".

Function blocks can also operate with temporary tags. Temporary tags are will not be stored in the instance DB, but are available for one cycle only.

Application

Function blocks contain subroutines that are always executed when a function block is called by another logic block. A function block can also be called several times at different points in a program. As a result, they simplify programming of frequently recurring functions.

Instances of function blocks

A call of a function block is referred to as an instance. The instance data is assigned to each instance of a function block with which the function block operates. The instance data can be declared in a data block (instance data block) or in the data of the calling block.

Access modes

S7-1200 offers to different access options for the instance data blocks, which can be assigned to a function block when this is called:

- Data blocks with optimized access

  Data blocks with optimized access have no fixed defined structure. The declaration elements contain only one symbolic name in the declaration, no fixed addressing within the block.

- Data blocks with standard access (compatible with S7-300/400)

  Data blocks with standard access have a fixed structure. The declaration elements contain both a symbolic name in the declaration and a fixed address within the block.

Note

To avoid errors when working with function blocks, refer to the section "Parameter transfer at block call (Page 707)".

See also

Creating functions and function blocks (Page 845)

Multi-instances (Page 706)

Instance data blocks (Page 699)

Basics of block access (Page 700)

Global data blocks (DB)

Definition

Data blocks are used to store program data. Data blocks thus contain variable data that is used by the user program. Global data blocks store data that can be used by all other blocks.

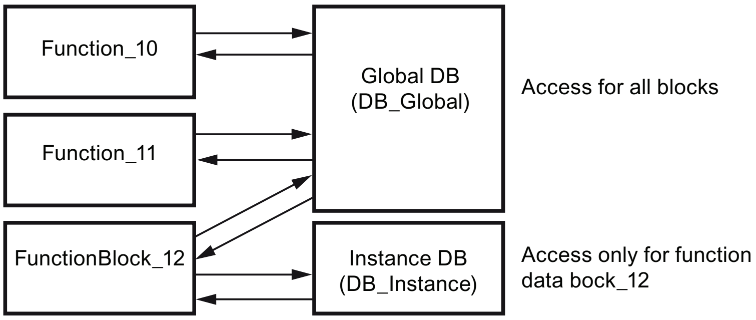
The maximum size of data blocks varies depending on the CPU. You can define the structure of global data blocks anyway you please.

You also have the option of using PLC data types (UDT) as a template for creating global data blocks.

Global data blocks in the user program

Every function block, function, or organization block can read the data from a global data block or can itself write data to a global data block. This data remains in the data block even after the data block is exited. A global data block and an instance data block can be open at the same time.

The following figure shows the different accesses to data blocks:



Access modes

S7-1200 offer two different access options for global data blocks:

- Data blocks with optimized access

  Data blocks with optimized access have no fixed defined structure. The declaration elements contain only one symbolic name in the declaration, no fixed addressing within the block.
- Data blocks with standard access

  Data blocks with standard access have a fixed structure. The declaration elements contain both a symbolic name in the declaration and a fixed address within the block.

See also

Creating data blocks (Page 846)

Basics of block access (Page 700)

Instance data blocks

Definition

The call of a function block is referred to as an instance. The data with which the instance works is stored in an instance data block.

The maximum size of instance data blocks varies depending on the CPU. The variables declared in the function block determine the structure of the instance data block.

Access modes

S7-1200 offers to different access options for the instance data blocks, which can be assigned to a function block when this is called:

- Data blocks with optimized access

  Data blocks with optimized access have no fixed defined structure. The declaration elements contain only one symbolic name in the declaration, no fixed addressing within the block.

- Data blocks with standard access

  Data blocks with standard access have a fixed structure. The declaration elements contain both a symbolic name in the declaration and a fixed address within the block.

See also: Call function blocks as single or multi-instances (Page 704)

See also

Creating data blocks (Page 846)

Basics of block access (Page 700)

Blocks with optimized access

Basics of block access

Introduction

S7-1200 provides data blocks with different access options:

- Data blocks with optimized access
- Data blocks with standard access

When you create a global data block or a function block, you can define the type of access of the global data block or the instance data block that will be assigned to the FB.

Within one program you can randomly combine the two types of blocks. You define the access type bindingly when you create the block. It is not possible to subsequently change the access type.

Data blocks with optimized access

Data blocks with optimized access have no fixed defined structure. The declaration elements contain only one symbolic name in the declaration, no fixed addressing within the block. The elements are saved automatically in the available memory area in such a way as to make optimal use of this area's capacity.

Variables in these data blocks can only be addressed in symbolic form. For example, you access the "Fill Level" variable in the "Data" DB as follows:

```
"Data".Fill Level
```

The optimized access offers the following advantages:

- The data are structured and stored in a way that is optimal for the CPU used. This allows you to increase the performance of the CPU.
- Access errors, from the HMI for example, are not possible.
- You can define specific individual variables as retentive.

Data blocks with standard access

Data blocks with standard access have a fixed structure. The declaration elements contain both a symbolic name in the declaration and a fixed address within the block. The address is shown in the "Offset" column.

Variables in these data blocks can be addressed in both symbolic and absolute form.

```
"Data".Fill Level
DB1.DBW2
```

Retentivity for optimized block access

In data blocks with optimized access you can define the retentive behavior of individual tags.

For structured data type tags, the retentivity setting always applies to the entire structure. You cannot make any individual retentivity setting for separate elements within the data type.

If a variable or structure is defined as retentive it is automatically stored in the retentive memory area of the data block.

Retentivity for Standard Access

In data blocks with standard access, you cannot set the retentive behavior of individual tags. The retentivity setting is valid for all variables of the data block.

9.1.1.3 Block calls

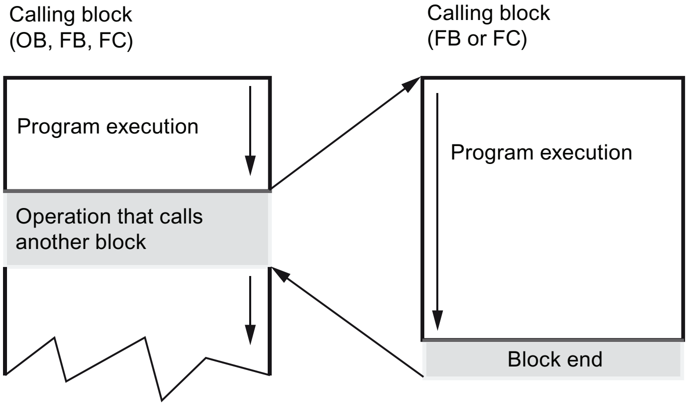
Principles of block calls

Function of block calls

For your blocks to be executed in the user program, they need to be called from another block.

When one block calls another block, the instructions of the called block are executed. Only when execution of the called block has been completed does execution of the calling block resume. The execution is continued with the instruction that follows on the block call.

The following figure shows the sequence of a block call within a user program:



Parameter transfer

When a block is called, you must assign values to the parameters in the block interface. By providing input parameters you specify the data with which the block is executed. By providing the output parameters you specify where the execution results are saved.

See also

Call hierarchy (Page 703)

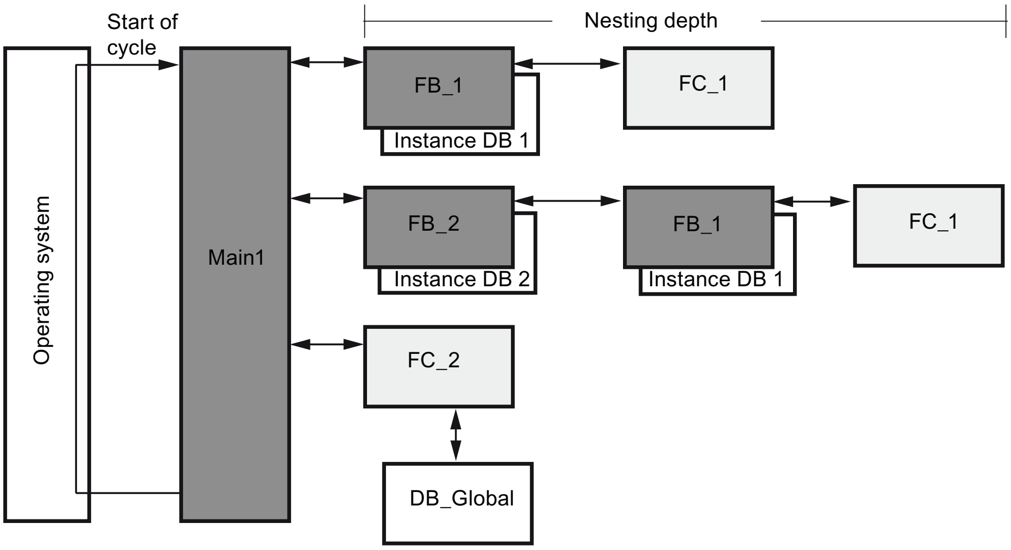
Principles for single instances and multi-instances (Page 704)

Call hierarchy

Definition

The order and nesting of block calls is referred to as the call hierarchy. The permissible nesting depth depends on the CPU.

The following figure shows an example of the order and nesting of block calls within an execution cycle:



See also

Principles for single instances and multi-instances (Page 704)

Principles of block calls (Page 702)

Call function blocks as single or multi-instances

Principles for single instances and multi-instances

Use of single instances and multi-instances

Function blocks (FBs) store their data in instance data blocks. Instance data blocks store the values of the block parameters and the static local data of the function blocks.

You can assign instance data blocks as follows:

- Single instance:

  One instance data block for each instance of a function block
- Multi-instance:
  - One instance data block for multi-instances of a function block
  - One instance data block for several instances of different function blocks

See also

Principles of block calls (Page 702)

Multi-instances (Page 706)

Single instances (Page 704)

Call hierarchy (Page 703)

Single instances

Definition

The call of a function block, which is assigned its own instance data block, is called a single instance data block.

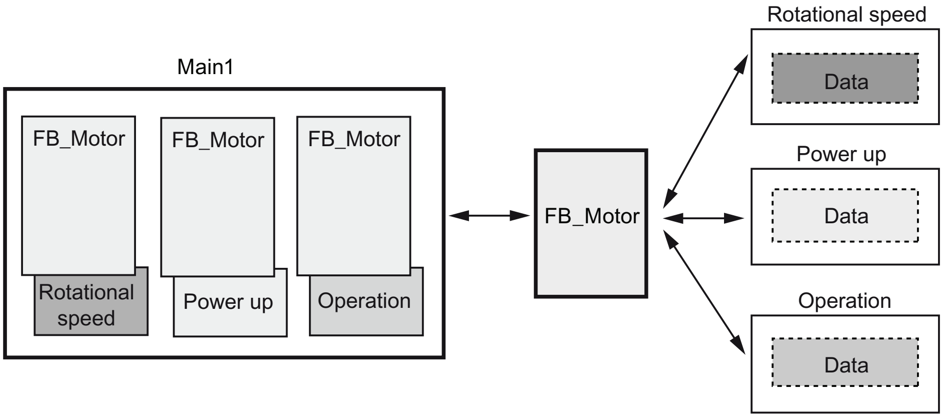
By assignment of the instance data block, you specify where the instance data of the FB is to be stored. By assigning a different instance data block to each call, you can use the same FBs several times with different instance data in each case.

Example of a single instance

You can control several motors using one function block. For this purpose, you assign a different instance data block for each function block call for motor control.

The different data for the various motors, such as speed, ramp-up time, total operating time, are saved in the different instance data blocks. A different motor will be controlled, depending on the instance data block assigned.

The following figure shows the control of three motors using one function block and three different data blocks:



See also

Principles for single instances and multi-instances (Page 704)

Multi-instances (Page 706)

Multi-instances

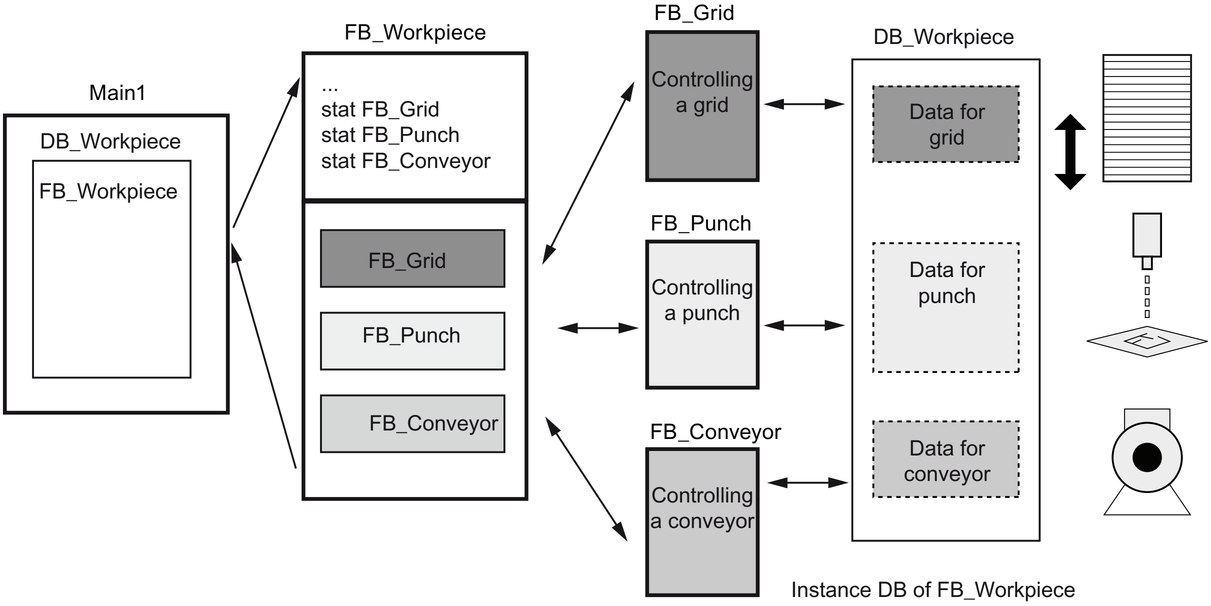
Definition

Multi-instances enable a called function block to store its data in the instance data block of the calling function block.

This allows you to concentrate the instance data in one instance data block and thus make better use of the number of instance data blocks available.

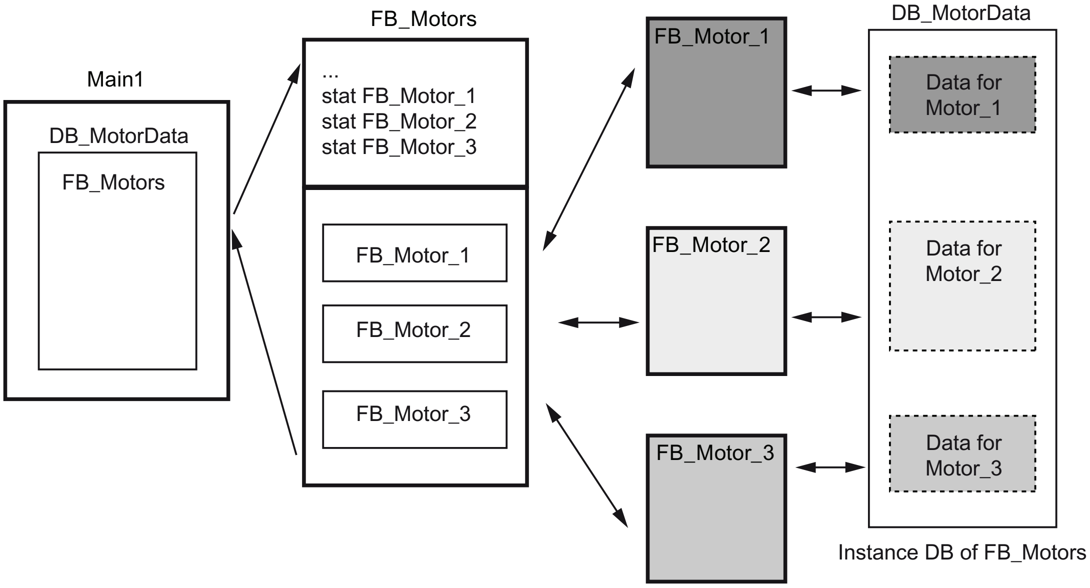
One instance data block for the instances of different function blocks

The following figure shows how multiple different function blocks store their data in a calling block. The FB_Workpiece calls the following on after the other: FB_Grid, FB_Punch and FB_Conveyor. The called blocks store their data in the DB_Workpiece, the instance data block of the calling block.



One instance data block for multi-instances of a function block

The following figure shows how a function block that is called in multi-instances stores the data for all the instances in one instance data block.



The function block FB_Motors calls three instances of the FB_Motor. The instances are "Motor_1", "Motor_2" and "Motor_3". Each call uses different instance data. However, all instance data are located in a single instance data block, DB_MotorData.

See also

Principles for single instances and multi-instances (Page 704)

Single instances (Page 704)

Parameter transfer at block call

Block parameters

Introduction

The calling block gives the called block the values with which it is to work. These values are referred to as block parameters. The input parameters provide the called block with the values that it has to process. The block returns the results via the output parameters.

Block parameters are therefore the interface between the calling and the call block.

You use input parameters when you want to only query or read values, and output parameters when you want to set or write these values. If block parameters are read and written you have to create these as in-out parameters.

Formal and actual parameters

The block parameters are defined in the interface of the called block. These parameters are referred to as formal parameters. They are placeholders for the parameters that are transferred to the block when it is called. The parameters transferred to the block when it is called are referred to as actual parameters.

See also

Parameter assignment to function blocks (Page 710)

Parameter assignment to functions (Page 709)

General rules for assigning parameters (Page 708)

General rules for assigning parameters

Rules for supplying block parameters with values

When you call a block with block parameters, assign actual parameters to its formal parameters. The following rules apply:

- Either a variable or a constant can be specified as an actual parameter.
- The actual parameter must be of the same data type as the formal parameter.
  - You can transfer structures as parameters. If a block has an input parameter of the STRUCT type, you must transfer as actual parameter a STRUCT with identical structure. You can also transfer individual elements of an STRUCT as actual parameter if the element corresponds to the data type of the formal parameter.
  - You can transfer ARRAYs as parameters. If a block has an input parameter of ARRAY type, you must transfer as actual parameter an ARRAY with identical structure. You can also transfer individual elements of an ARRAY as actual parameter if the element corresponds to the data type of the formal parameter.
  - You can transfer variables that have a PLC data type as parameters. If the parameter is declared as the PLC data type in the variable declaration, you must transfer a data type that has the same data element structure. However, an element of a PLC data type can be assigned to a parameter when a block is called, provided that the element of the PLC data type matches the data type of the parameter.

See also

Parameter assignment to function blocks (Page 710)

Parameter assignment to functions (Page 709)

Block parameters (Page 707)

Parameter assignment to functions

Supplying parameters of functions (FC)

Functions have no data memory in which values of block parameters can be stored. Therefore, when a function is called, all formal parameters must be assigned actual parameters.

Function Value

Functions normally calculate a function value. This function value can be returned to the calling block via the RET_VAL output parameter. For this, the RET_VAL output parameter must be declared in the interface of the function. RET_VAL is always the first output parameter of a function. All data types are permitted for the RET_VAL parameter except ARRAY and STRUCT, as well as parameter types TIMER and COUNTER.

In the SCL programming language functions can be call directly in an expression. The result of the expression is then formed with the calculated function value. Therefore, the data type ANY is also not permitted in SCL for the function value.

See also

Parameter assignment to function blocks (Page 710)

Block parameters (Page 707)

General rules for assigning parameters (Page 708)

Calling functions (Page 997)

Examples for calling functions in SCL (Page 999)

Parameter assignment to function blocks

Supplying parameters of function blocks (FB)

In the case of function blocks the parameter values will be stored in the instance data.

If the input, output, or in-out parameters of a function block were not assigned with values, the stored values are used. In some cases, you must assign parameters.

The following table shows which parameters of a function block must be assigned actual parameters:

Parameter	Elementary data type	Structured data type	Parameter type
Input (Input)	optional	optional	required
Output (Output)	optional	optional	required
In-out (InOut)	optional	required	Permitted with S7-1200 only, parameter assignment required

Assigning default values to formal parameters

You can assign default values to formal parameters in the function block interface. These values are transferred to the associated instance data.

If you do not assign actual parameters to the formal parameters in the call instruction, the values currently stored in the instance data block will be used.

The following table shows which variables can be assigned a default value:

Variables	Elementary data type	Structured data type	Parameter type
Input (Input)	optional	optional	not possible
Output (Output)	optional	optional	not possible
In-out (InOut)	optional	not possible	not possible
Static (Static)	optional	optional	not possible
Temporary (Temp)	not possible	not possible	not possible

See also

Block parameters (Page 707)

General rules for assigning parameters (Page 708)

Parameter assignment to functions (Page 709)

Forwarding of block parameters

Basic information on forwarding block parameters

Introduction

The "Forwarding" of block parameters is a special type of parameter assignment. In this case the block parameters of the calling block are forwarded to the parameters of the called block. The called block uses the values that are currently present at the block parameters of the calling block as the actual parameters.

The following general rules apply to this:

- Both block parameters must be of the same data type.
- Input parameters can only be forwarded at the input parameter.
- Output parameters can only be forwarded at output parameters.
- In-out parameters can be forwarded at all parameter types.

Detailed rules are described in the following sections.

See also

Calling a function by another function (Page 712)

Calling a function by a function block (Page 713)

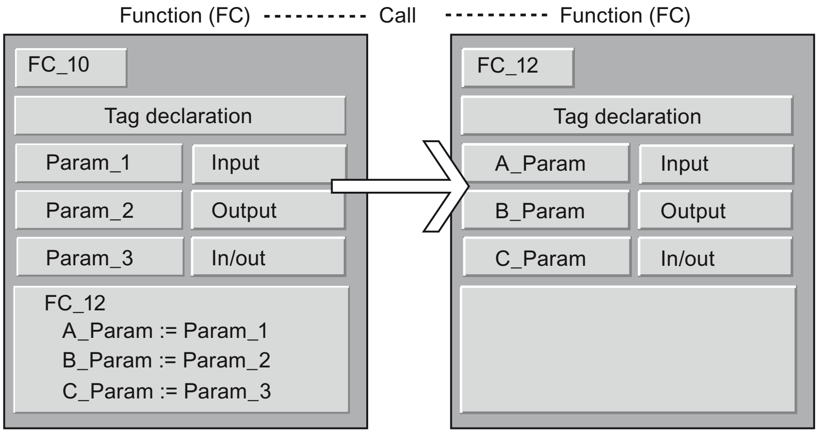
Calling a function block by a function (Page 714)

Calling a function block by another function block (Page 715)

Calling a function by another function

Permissible data types for the call of a function by another function

You can forward the formal parameters of the calling function to the formal parameters of the called function. The following figure shows the formal parameters of the function FC_10, which are forwarded to the formal parameters of the function FC_12:



Specific rules apply to the forwarding of formal parameters. The following table shows the rules that apply when one function calls another function:

Calling FC → Called FC	Elementary data types	Structured data types	Parameter types
Input -> Input	S7-300/400 S7-1200	S7-1200	-
Output -> Output	S7-300/400 S7-1200	S7-1200	-
In-out -> Input	S7-300/400 S7-1200	S7-1200	-
In-out -> In-out	S7-300/400 S7-1200	S7-1200	-
In-out -> Output	S7-300/400 S7-1200	S7-1200	-

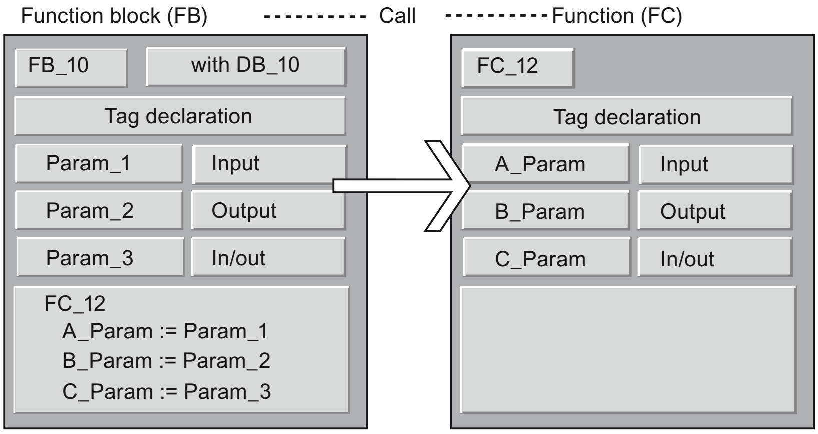
See also

Basic information on forwarding block parameters (Page 711)

Calling a function by a function block

Permissible data types for the call of a function by a function block

You can forward the formal parameters of the calling function block to the formal parameters of the called function. The following figure shows the formal parameters of the function block FB_10, which are forwarded to the formal parameters of the function FC_12:



Specific rules apply to the forwarding of formal parameters. The following table shows the rules that apply when a function block calls a function:

Calling FB → Called FC	Elementary data types	Structured data types	Parameter types
Input -> Input	S7-300/400 S7-1200	S7-300/400 S7-1200	-
Output -> Output	S7-300/400 S7-1200	S7-300/400 S7-1200	-
In-out -> Input	S7-300/400 S7-1200	S7-1200	-
In-out -> Output	S7-300/400 S7-1200	S7-1200	-
In-out -> In-out	S7-300/400 S7-1200	S7-1200	-

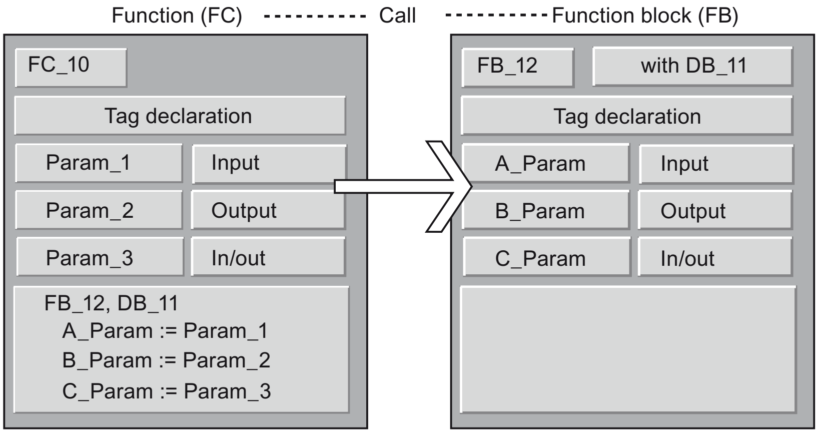
See also

Basic information on forwarding block parameters (Page 711)

Calling a function block by a function

Permissible data types for the call of a function block by a function

You can assign the formal parameters of the calling function to the formal parameters of the called function block. The following figure shows the formal parameters of the function FC_10, which are forwarded to the formal parameters of the function block FB_12:



Specific rules apply to the forwarding of formal parameters. The following table shows the rules that apply when a function calls a function block:

Calling FC → Called FB	Elementary data types	Structured data types	Parameter types
Input -> Input	S7-300/400 S7-1200	S7-1200	S7-300/400
Output -> Output	S7-300/400 S7-1200	S7-1200	-
In-out -> Out	S7-300/400 S7-1200	S7-1200	-
In-out -> Input	S7-300/400 S7-1200	S7-1200	-
In-out -> In-out	S7-300/400 S7-1200	S7-1200	-

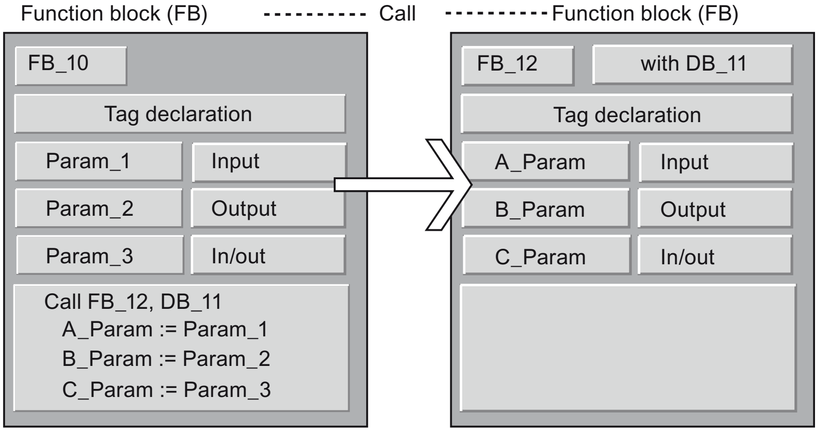
See also

Basic information on forwarding block parameters (Page 711)

Calling a function block by another function block

Permissible data types for the call of a function block by another function block

You can forward the formal parameters of the calling function block to the formal parameters of the called function block. The following figure shows the formal parameters of the function block FB_10, which are forwarded to the formal parameters of the function block FB_12:



Specific rules apply to the forwarding of formal parameters. The following table shows the rules that apply when one function block calls another function block:

Calling FB → Called FB	Elementary data types	Structured data types	Parameter types
Input -> Input	S7-300/400 S7-1200	S7-300/400 S7-1200	S7-300/400
Output -> Output	S7-300/400 S7-1200	S7-300/400 S7-1200	-
In-out -> Input	S7-300/400 S7-1200	S7-1200	-
In-out -> Out	S7-300/400 S7-1200	S7-1200	-
In-out -> In-out	S7-300/400 S7-1200	S7-1200	-

See also

Basic information on forwarding block parameters (Page 711)

9.1.1.4 Operands in instructions

Basic information about operands

Introduction

When you program instructions you must specify which data values the instruction should process. These values are referred to as operands. You can, for example, use the following elements as operands:

- PLC variables
- Constants
- Variables in instance data blocks
- Variables in global data blocks

Absolute address and symbolic name

Operands are identified by means of an absolute address and a symbolic name. You define the names and addresses in the PLC variable table or in the variable declaration of the blocks.

Data blocks with optimized access (S7-1200)

Data elements in data blocks with optimized access only receive a symbolic name and no absolute address in the declaration. For more information on this, refer to "See also".

See also

Displaying symbolic and absolute addresses (Page 896)

Basics of block access (Page 700)

Using tags within the program

Definition

A variable is a placeholder for a data value that can be changed in the program. The format of the data value is defined. The use of variables makes your program more flexible. For example, you can assign different values to variables that you have declared in the block interface for each block call. As a result, you can reuse a block you have already programmed for various purposes.

A variable consists of the following elements:

- Name
- Data type
- Absolute address
  - PLC tags and DB tags in blocks with standard access have an absolute address.
  - DB variables in blocks with optimized access have no absolute address.
- Value (optional)

Declaring Variables

You can define variables with different scopes for your program:

- PLC tags that apply in all areas of the CPU
- DB variables in global data block that can be used by all blocks throughout the CPU.
- DB tags in instance data blocks that are predominantly used within the block in which they are declared.

The following table shows the difference between the variable types:

	PLC tags	Variables in instance DBs	Variables in global DBs
Range of application	• Are valid throughout the entire CPU. • Can be used by all blocks on the CPU. • The name is unique within the CPU.	• Are predominantly used in the block in which they are defined. • The name is unique within the instance DB.	• Can be used by all blocks on the CPU. • The name is unique within the global DB.
Permissible characters	• Letters, numbers, special characters • Quotation marks are not permitted. • Reserved keywords are not permitted.	• Letters, numbers, special characters • Reserved keywords are not permitted.	• Letters, numbers, special characters • Reserved keywords are not permitted.

	PLC tags	Variables in instance DBs	Variables in global DBs
Use	• I/O signals (I, IB, IW, ID, Q, QB, QW, QD) • Bit memory (M, MB, MW, MD)	• Block parameters (input, output and in-out parameters), • Static data of a block • Temporary local data of a block	• Static data
Location of definition	PLC tag table	Block interface	Declaration table of the global DB

See also

Reserved key words (Page 719)

Basic information about operands (Page 716)

Displaying symbolic and absolute addresses (Page 896)

Valid names of PLC tags (Page 829)

Permissible addresses and data types of PLC tags (Page 829)

Constants

Definition

A constant defines an unchangeable data value. Constants can be read by various program elements during the execution of the program but cannot be overwritten. A change of the constant value during the program's execution can lead to syntax or runtime errors.

Symbolic constants (S7-1200)

In S7-1200 you have the option of declaring symbolic names for constants and thus making static values available under a name in the program. The symbolic constants are valid throughout the CPU. Constants are declared in the "Constants" tab of the PLC variable table.

Elements of Constants

A constant consists of the following elements:

- Name (in the case of symbolic constants)

  Valid characters in the constant name are letters, number and special characters; invalid characters are quotation marks and reserved keywords.
- Data type
- Constant value

  The input format and the value range of the constant depend on the data type of the constant.

More information

For more information on data types of constants and their input formats and value ranges, refer to the "Data types" section under "See also".

See also

Rules for symbolic constants (Page 835)

Enter constants (Page 739)

Declaring constants (Page 835)

Reserved key words

SIMATIC recognizes a range of key words whose definitions are fixed and which have a certain meaning in the program. You should not use these keywords as names for tags or constants.

If it necessary to use a keyword as a tag name, enclose it in quotation marks or append an # character in front of it.

Table of reserved key words

The following table shows all the reserved key words.

Keywords German mnemonics	Keywords English mnemonics	Description
A	Q	Output, bit
A1	CC1	Condition code bit
A0	CC0	Condition code bit
AB	QB	Output, byte
AD	QD	Output, double word
ANY	ANY	Designation for ANY data type
AR1	AR1	Address Register 1
AR2	AR2	Address Register 2

Keywords German mnemonics	Keywords English mnemonics	Description
ARRAY	ARRAY	Introduces the specification of an array and is followed by the index list between "[" and "]"
AUTHOR	AUTHOR	Name of the author, company name, department name, or other name (max. 8 characters, no spaces)
AW	QW	Output, word
AX	QX	Output, bit
B	B	Byte
BEGIN	BEGIN	Introduces the instruction part for logic blocks or initialization part for a data block
BIE	BR	Binary result
BOOL	BOOL	Elementary data type for binary data
BY	BY	Increment of the FOR loop
BYTE	BYTE	Elementary data type
CALL	CALL	Call
CASE	CASE	Introduction to the CASE statement
CHAR	CHAR	Elementary data type
CODE_VERSION1	CODE_VERSION1	Label, whether an FB is multi-instance capable or not. If you want to declare multi-instances, the FB must not have this characteristic.
CONST	CONST	Start of the constant declaration
CONTINUE	CONTINUE	Instruction to exit a loop in SCL
DATA_BLOCK	DATA_BLOCK	Introduces the data block
DATE	DATE	Elementary data type for date
DATE_AND_TIME	DATE_AND_TIME	Complex data type for date and time of day
DB	DB	Data block
DBB	DBB	Data block, data byte
DBD	DBD	Data block, data double word
DBLG	DBLG	Data block length
DBNO	DBNO	Data block number
DBW	DBW	Data block, data word
DBX	DBX	Data block, data bit
DI	DI	Instance data block
DIB	DIB	Instance data block, data byte
DID	DID	Instance data block, data double word
DILG	DILG	Instance data block length

Keywords German mnemonics	Keywords English mnemonics	Description
DINO	DINO	Instance data block number
DINT	DINT	Elementary data type for whole numbers (integers) double precision
DIW	DIW	Instance data block, data word
DIX	DIX	Instance data block, data bit
DO	DO	Introduction of the instruction part in FOR and WHILE instruction
DT	DT	Elementary data type for date and time
DWORD	DWORD	Elementary data type for double word
E	I	Input (via process image), bit
EB	IB	Input (via process image), byte
ED	ID	Input (via process image), double word
ELSE	ELSE	Alternative branch in IF and CASE statement
ELSIF	ELSIF	Alternative condition of the IF instruction
EN	EN	System operand of the EN/ENO mechanism
ENO	ENO	System operand of the EN/ENO mechanism
END_CASE	END_CASE	End of the CASE statement
END_DATA_BLOCK	END_DATA_BLOCK	Ends the data block
END_FOR	END_FOR	End of the FOR statement
END_FUNCTION	END_FUNCTION	Ends the function
END_FUNCTION_BLOCK	END_FUNCTION_BLOCK	Ends the function block
END_IF	END_IF	End of the IF instruction
END_ORGANIZATION_BLOCK	END_ORGANIZATION_BLOCK	Ends the organization block
END_REPEAT	END_REPEAT	End of the REPEAT statement
END_STRUCT	END_STRUCT	Ends the specification of a structure
END_SYSTEM_FUNCTION	END_SYSTEM_FUNCTION	Ends the system function
END_SYSTEM_FUNCTION_BLOCK	END_SYSTEM_FUNCTION_BLOCK	Ends the system function block
END_TYPE	END_TYPE	Ends the UDT
END_VAR	END_VAR	Ends a declaration block
END_WHILE	END_WHILE	End of the WHILE instruction
EW	IW	Input (via process image), word
EXIT	EXIT	Instruction to exit a loop in SCL

Keywords German mnemonics	Keywords English mnemonics	Description
FALSE	FALSE	Predefined Boolean constant: Logical condition false, value equal to 0
FAMILY	FAMILY	Block family name: for example, closed-loop controller (max. 8 characters, no spaces)
FB	FB	Function block
FC	FC	Function
FOR	FOR	Introduction of the FOR statement
FUNCTION	FUNCTION	Introduces the function
FUNCTION_BLOCK	FUNCTION_BLOCK	Introduces the function block
GOTO	GOTO	Introduction of the GOTO statement
IF	IF	Introduction of the IF instruction
INT	INT	Elementary data type for whole numbers (integers) single precision
KNOW_HOW_PROTECT	KNOW_HOW_PROTECT	Block protection; a block compiled with this option permits no view of the instruction part.
L	L	Local data bit
LB	LB	Local data byte
LD	LD	Local data double word
LW	LW	Local data word
M	M	Memory bit
MB	MB	Memory byte
MD	MD	Memory double word
MOD	MOD	Modulo operator
MW	MW	Memory word
NAME	NAME	Block name (max. 8 characters)
NETWORK	NETWORK	Network
NOT	NOT	Logic inversion
OB	OB	Organization block
OF	OF	Introduction of the data type specification / Introduction of the instruction part of the CASE statement
OR	OR	Or logical operation of logical expressions
ORGANIZATION_BLOCK	ORGANIZATION_BLOCK	Introduces the organization block
OS	OS	Save overflow
OV	OV	Overflow

Keywords German mnemonics	Keywords English mnemonics	Description
PA	PQ	Output (direct peripherals), bit
PAB	PQB	Output (direct peripherals), byte
PAD	PQD	Output (direct peripherals), double word
PAW	PQW	Output (direct peripherals), word
PE	PI	Input (direct peripherals), bit
PEB	PIB	Input (direct peripherals), byte
PED	PID	Input (direct peripherals), double word
PEW	PIW	Input (direct peripherals), word
POINTER	POINTER	Pointer data type, only permitted in parameter declaration in the parameter block
READ_ONLY	READ_ONLY	Write protection for data blocks; their data can be read, but not changed.
REAL	REAL	Elementary data type
REPEAT	REPEAT	Introduction of the REPEAT statement
RET_VAL	RET_VAL	Return value
RETURN	RETURN	RETURN statement in SCL
S5T	S5T	Syntax for S5 data type
S5TIME	S5TIME	Elementary data type for time information, special S5 format
S7_	S7_	Keywords for system attributes
SDB	SDB	System data block
SFB	SFB	System function block
SFC	SFC	System function
STRING	STRING	Data type for character string
STRUCT	STRUCT	Introduces the specification of a structure and is followed by a list of components
STW	STW	Status word
SYSTEM_FUNCTION	SYSTEM_FUNCTION	System function
SYSTEM_FUNCTION_BLOCK	SYSTEM_FUNCTION_BLOCK	System function block
T	T	Time element (timer)
THEN	THEN	Introduction of the instruction part of an IF instruction
TIME	TIME	Elementary data type for time information
TIME_OF_DAY	TIME_OF_DAY	Elementary data type for time of day
TITLE	TITLE	Optional block title or network title

Keywords German mnemonics	Keywords English mnemonics	Description
TO	TO	Definition of the full-scale value of a FOR statement
TOD	TOD	Elementary data type for time of day
TRUE	TRUE	Predefined Boolean constant: Logical condition true, value not equal to 0
TYPE	TYPE	Introduces UDT
UDT	UDT	Global or PLC data type
UNLINKED	UNLINKED	Marking 'non runtime-related'
UNTIL	UNTIL	End of the instruction part of a REPEAT statement
UO	AO	Query after (Q1=1) AND (Q0=1)
VAR	VAR	Introduces a declaration block
VAR_IN_OUT	VAR_IN_OUT	Introduces a declaration block
VAR_INPUT	VAR_INPUT	Introduces a declaration block
VAR_OUTPUT	VAR_OUTPUT	Introduces a declaration block
VAR_TEMP	VAR_TEMP	Introduces a declaration block
VERSION	VERSION	Version number of the block
VOID	VOID	Function has no return value
WHILE	WHILE	Introduction of a WHILE instruction
WORD	WORD	Elementary data type for word
XOR	XOR	Logic operation
Z	C	Counter
$_<any character>	$_	Alignment symbol

Addressing operands

Addressing global variables

Addressing global variables

To address a global PLC variable, you can use the absolute address or the symbolic name.

Addressing global variables in symbolic form

When you use addressing in symbolic form, you enter the variable name from the PLC variable table. The symbolic name of global variables are automatically enclosed in quotation marks.

Addressing global variables in absolute form

When you use addressing in absolute form, you enter the address of the variables from the PLC variable table. The absolute address uses numerical addresses starting with zero for each operand range.

Examples

The following examples show applications of symbolic and absolute addressing:

Addressing	Explanation
Q1.0	Absolute address: Output 1.0
I16.4	Absolute address: Input 16.4
IW4	Absolute address: Input word 4
"Motor"	Symbolic address "Motor"
"Value"	Symbolic address "Value"

See also: Permissible addresses and data types of PLC tags (Page 829)

See also

Displaying symbolic and absolute addresses (Page 896)

Accessing I/O devices (Page 725)

Accessing I/O devices

Description

The process image of the CPU is updated once in a cycle. In time-critical applications, however, it can be that the current state of a digital input or output has to be read or transferred more often than once per cycle. For this purpose you can use a suffix for I/O access identifiers on the operand to directly access the I/O.

If you want to read the input directly from the peripherals, use the peripheral inputs memory area (PI) instead of the process input image (I). The peripherals memory area can be read as a bit, byte, word, or double word.

If you want to write directly to the output, use the peripheral output (PQ) memory area instead of the process output image (Q). The peripheral output memory area can be written as a bit, byte, word, or double word.

To read or write a signal directly from a peripheral input, you can add the suffix for I/O access ":P", to the operand.

⚠ WARNING

Immediate writing to the I/O can lead to hazardous states, for example when writing multiple times to an output in one program cycle.

Syntax

```
<Operand>:P
```

Example

The following example shows applications of I/O access identifiers:

Addressing	**Description**
"Motor"	//Addresses the variable "Motor" in the process input image.
"Motor":P	//Addresses the variable "Motor" in the memory area of the peripheral inputs (PI).
"Switch"	//Addresses the variable "Switch" in the process output image.
"Switch":P	//Addresses the variable "Switch" in the memory area of the peripheral outputs (PQ).

See also

Addressing global variables (Page 724)

Addressing variables in data blocks

Addressing variables in global data blocks

Description

Variables in global data blocks can be addressed in symbolic or absolute form. For symbolic addressing, you use the name of the data block and the name of the variable, separated by a dot. The name of the data block is enclosed in quotation marks.

For absolute addressing, you use the number of the data block and the absolute address of the variables in the data block, separated by a dot.

Note

Variables in blocks with optimized access can only be addressed in symbolic form.

Syntax

```
"<DBname>".VariableName
<DBnummer>.absoluteAddress
```

The following table show the possible absolute addresses of variables in data blocks:

Data type	Absolute address	Example	Description
BOOL	DBn.DBXx.y	DB1.DBX1.0	Data bit 1.0 in DB1
BYTE, CHAR, SINT, USINT	DBn.DBBy	DB1.DBB1	Data bit 1 in DB1
WORD, INT, UINT	DBn. DBWy	DB1.DBW1	Data word 1 in DB1
DWORD, DINT, UDINT, REAL, TIME	DBn.DBDy	DB1.DBD1	Data double word 1 in DB1

Example

The following examples show the addressing of variables in global data blocks:

Addressing	Description
"Motor".Value	Symbolic addressing of the "Value" variable in the "Motor" global data block.
DB1.DBX1.0	Absolute addressing of the "DBX1.0" tags in the "DB1" global data block.

See also

Addressing structured variables (Page 729)

Addressing individual bits of a variable (Page 730)

Basics of indirect addressing (Page 732)

Addressing instance data (Page 728)

Addressing instance data

Description

You can address data elements from the interface of the current block. These variables are stored in the instance data block. It is not possible to address variables from external instance data blocks.

Note

Variables in blocks with optimized access can only be addressed in symbolic form.

To address a variable from the interface of the current block, enter the character # followed by the symbolic variable name.

You can also access the variables of a multi-instance block. Within the multi-instance block, also use the character # followed by the variable name to address the data. You access the data of the multi-instance block from the calling block using "Multi-instanceName.VariableName".

Syntax

Use the following syntax for addressing variables in instance data blocks:

```
#<VariableName>
#<Multi-instanceName.VariableName>
```

Examples

The following examples show the addressing of variables in instance data blocks:

Addressing	Description
#Value	Addressing the "Value" variable in the instance data block.
#On	Addressing the "On" variable within the multi-instance block
#Multi.On	Addressing the "On" variable of the multi-instance block from the calling block

See also

Addressing variables in global data blocks (Page 726)

Addressing structured variables (Page 729)

Addressing individual bits of a variable (Page 730)

Basics of indirect addressing (Page 732)

Addressing structured variables

Addressing data elements of an ARRAY

You access an element in an ARRAY using the index of the element in the ARRAY. An index consists of any integer value (-32768 to 32767) enclosed in square brackets. The index has one value per dimension.

See also:

Array (Page 765)

Indirect indexing of ARRAY components (Page 734)

Addressing data elements in structures

You access individual elements in a structure using "StructureName.ElementName".

See also:

Structures (Page 768)

Addressing data elements of an PLC data type

The syntax `PLCDataTypeName.ElementName` is used to access PLC data types in the program.

See also:

PLC data types (Page 775)

Example:

The following examples show the addressing of structured data type variables:

Addressing	Description
Motor.Value_1x3[2]	Addressing of a one-dimensional array
Motor.Value_2x4[2,4]	Addressing of a two-dimensional array
Motor.Value_4x7[2,4,1,3]	Addressing of a four-dimensional array
Batch_1.Temperature	Addressing of the element "Temperature" in the structure "Batch_1"
Values.Temperature	Addressing of the element "Temperature" in the PLC data type "Values"

See also

Basics of indirect addressing (Page 732)

Addressing individual bits of a variable

Description

You have the option to specifically address areas within declared tags. You can access width areas 1 bit, 8 bit or 16 bit.

Syntax

The following syntax is used for addressing:

```
<Tag>.X<Bit number>
<Tag>.B<Byte number>
<Tag>.W<Word number>
```

The syntax has the following components:

Part	Description
<Tag>	Tag that you access. The tag must be of the "Bit string" data type. In the case of activated IEC check, the access to tags of the "Integer" data type is also possible.
X B W	ID for the access width "Bit (1Bit)" ID for the access width "Byte (8 Bit)" ID for the access width "Word (16 Bit)"
<Bit number>	Bit number within <tag> that is accessed. The number 0 accesses the lowest bit.
<Byte number>	Byte number within <tag> that is accessed. The number 0 accesses the lowest byte.
<Word number>	Word number within <tag> that is accessed. The number 0 accesses the lowest word.

Examples

The following examples show the addressing of individual bits:

Addressing	**Description**

`"Engine".Motor.X0`
`"Engine".Motor.X7`
	`"Motor" is a tag of the BYTE data type in the global data block "Engine".`
	`X0 addresses the first bit, X7 the eighth bit within "Motor".`

`"Engine".Speed.B0`
`"Engine".Speed.B1`
	`"Speed" is a tag of the WORD data type in the global data block "Engine".`
	`B0 addresses the first bit, B1 the second byte within "Speed".`

`"Engine".Fuel.W0`
`"Engine".Fuel.W1`
	`"FUEL" is a tag of the DWORD data type in the global data block "Engine".`
	`W0 addresses the first word, W1 the second word within "Fuel".`

Overlaying tags with AT

Description

To access data areas within a declared tag, you can overlay the declared tags with an additional declaration. This provides you with the option of addressing an already declared tag with a different data type. You can, for example, address the individual bits of a tag of WORD data type with an ARRAY of BOOL.

Rules

The following rules apply to the overlaying of tags:

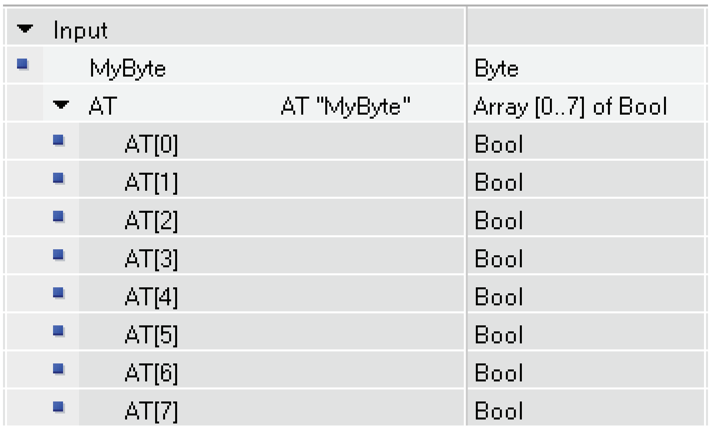
- Overlaying is only possible in S7-1200. The only exception is SCL. In this case overlaying is permitted in all CPU families.
- The overlaying of tags is only possible in the interface of code blocks with standard access.
- Tags from all code block types and all declaration sections (Input, Output, InOut, Static, Temp) can be overlaid.
- An overlaying tag can be used like any other tag in the block.
- It is not possible to overlay tags of data type VARIANT.
- The overlaying tag must be equal to or smaller than the tag that is overlaid.
- The declaration of an overlaying tag must take place immediately after the declaration of the tag to which it is to point and in the same declaration section.

Declaration

To overlay a tag, declare an additional tag directly after the tag that is to be overlaid and identify it with the keyword "AT".

Example

The following figure shows the declaration of an overlaid tag:



▼ Input		
MyByte		Byte
▼ AT	AT "MyByte"	Array [0..7] of Bool
AT[0]		Bool
AT[1]		Bool
AT[2]		Bool
AT[3]		Bool
AT[4]		Bool
AT[5]		Bool
AT[6]		Bool
AT[7]		Bool

When a block is called with the shown tag declaration, the "MyByte" tag is assigned. Within the block there are now two options for interpreting the data:

- as a byte
- As one-dimensional ARRAY of BOOL

See also

Declaring higher-level tags (Page 1022)

Addressing operands indirectly

Basics of indirect addressing

Introduction

In the case of indirect addressing, you calculate the used operands in runtime. With indirect addressing you can also execute program sections several times and use a different operand during each run.

The following general options of indirect addressing are permitted in all programming languages:

- Indirect addressing via pointer
- Indirect indexing of ARRAY components

Additional language-specific options are described in the following sections.

> ⚠ **CAUTION**
>
> Since the operands are only calculated in runtime in the case of indirect addressing, memory areas can be accidentally overwritten. Therefore, use indirect addressing only with caution.

See also

Indirect addressing in SCL (Page 735)

Indirect addressing in STL (Page 737)

Indirect addressing via pointer (Page 733)

Indirect indexing of ARRAY components (Page 734)

Indirect addressing via pointer

Description

For indirect addressing, a special data format is required that contains the address and possibly also the range and the data type of an operand. This data format is referred to as pointer. The following types of pointer are available to you:

- POINTER (S7-300/400)
- ANY (S7-300/400)
- VARIANT (S7-1200)

For more information on the pointer data type, refer to "See also".

Note

In SCL the use of the pointer data type is restricted. The only option available is to forward it to the called block.

Example

The following example shows an indirect addressing with an area-internal pointer:

Addressing in STL	Description
`L P#10.0`	`// Load pointer (P#10.0) in accumulator 1`
`T MD20`	`// Transfer pointer to the operand MD20`
`L MW [MD20]`	`// Load MW10 in accumulator 1`
`....`	`// Any program`
`L MD [MD20]`	`// Load MD10 in accumulator 1`
`....`	`// Any program`
`= M [MD20]`	`// If RLO=1, set the memory bit M10.0`

The pointer P#10.0 is transferred to the operand MD20. If the operand MD20 in square brackets is programmed, this will be replaced in runtime by the address that is contained in the pointer.

See also

Basics of indirect addressing (Page 732)

Indirect indexing of ARRAY components

Description

For addressing the components of an ARRAY, you can enter tags of the integer data type in addition to constants. When using tags the index will be calculated during runtime. You can, for example, use a different index for each cycle in program loops.

Syntax

The following syntax is used for the indirect indexing of a ARRAY:

```
"<Data block>".<ARRAY>["i"] // one-dimensional ARRAY
"<Data block>".<ARRAY>["i"] // one-dimensional ARRAY of STRUCT
"<Data block>".<ARRAY>["i"] // multidimensional ARRAY
"<Data block>".<ARRAY>["i"] // multidimensional ARRAY of STRUCT
```

The syntax has the following components:

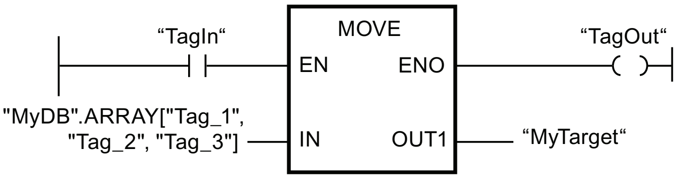
Part	Description
Data block	Name of the data block in which the ARRAY is located
ARRAY	Tag of the ARRAY data type
i, j	PLC tags of the integer data type that are used as pointers
a	Additional partial tag of the structure

Examples

The following examples show the indirect indexing of an ARRAY component using SCL as an example. MOTOR is a one-dimensional ARRAY_of_INT with three rows. VALUES is a PLC tag of the "Integer" data type.

Addressing in SCL	**Description**
`MOTOR[2] := VALUES;`	`(* direct addressing: Assignment of VALUES to the second row of the ARRAY MOTOR*)`
`MOTOR["Tag_1"] := VALUES;`	`(* Indirect addressing: Assignment of VALUES to the row specified by "Tag_1" of the ARRAY MOTOR*)`

The following example shows the indirect indexing of an ARRAY component as an example of LAD. ARRAY is a three-dimensional ARRAY. "Tag_1", "Tag_2" and "Tag_3" are PLC tags of the "Integer" data type. Depending on their values, one of the ARRAY components will be copied to the "MyTarget" tag.



More information

For more information on the ARRAY data type, refer to "See also".

See also

Basics of indirect addressing (Page 732)

Addressing structured variables (Page 729)

Indirect addressing in SCL

With SCL, you have the option of indirectly addressing the following objects:

- Global tags
- Tags in data blocks
- Data blocks

Indirect addressing of tags

Indirect addressing is performed similarly to absolute addressing. An offset in round brackets is specified instead of the address. The offset consists of one byte tag, in the case of operands of one byte and one bit tag. Byte and bit tags must be of data type INT.

Timers and counters from the PLC tag table cannot be addressed indirectly in this way.

Indirect addressing of data blocks

In addition to indexed access to DB tags, there is the option of using the conversion function WORD_TO_BLOCK_DB to address data block indirectly. The DB number is therefore specified as the tag or expression with WORD data type.

Syntax

The following syntax is used for the indirect indexing of data operands in SCL:

- Addressing of a global tag:

  Operand ID (Byte tag)

  Operand ID (Byte tag.Bit tag)
- Addressing of a DB tag:

  MyDB(Byte tag)

  MyDB(Byte tag.Bit tag)
- Addressing a data block

  WORD_TO_BLOCK_DB(Index).Operand ID (Address)

Examples

The following examples show indirect addressing with SCL:

Addressing in SCL

```
(*Depending on the value of the runtime tag i, an input word will be set to 0.*)

#i:=2
FOR #i := 2 TO 8 DO
 EW(#i) := 0  ;
END_FOR;
```

Addressing in SCL

```
 (*Depending on the value of the runtime tag i, a word in MyDB will be set to 0.*)

#i:=2
FOR #i := 2 TO 8 DO
 MyDB.DW(#i) := 0  ;
END_FOR;
```

Addressing in SCL

```
 (*indirect addressing using the WORD_TO_BLOCK_DB conversion function: Global tag
"Address index" of WORD data type is used as the number of the DB. *)

M0.0:=WORD_TO_BLOCK_DB("Adressindex").DX(0.0);
MW0:=WORD_TO_BLOCK_DB("Adressindex").DW(4);
```

Addressing in SCL
`(*indirect addressing using the WORD_TO_BLOCK_DB conversion function: Global tag "Address index" of WORD data type is used as the number of the DB. The data element within the DB is also specified via an index*)`

`M0.0:=WORD_TO_BLOCK_DB("Adressindex").DX(#i.#y);`
`MW0:=WORD_TO_BLOCK_DB("Adressindex").DW(#y);`

See also

Basics of indirect addressing (Page 732)

Indirect addressing in STL (Page 737)

Addressing structured variables (Page 729)

Indirect addressing in STL

Basic information about address registers

Introduction

Two address registers are available for the indirect addressing of operands: address register 1 (AR1), and address register 2 (AR2). The address registers are equal and are 32 bits in length. You can store area-internal and cross-area pointers in the address registers. To define the address of an operand, you can call the stored data in the program.

Data is exchanged between the registers and the other available memory areas with the assistance of load and transfer instructions.

For more information on the statements that address registers use and on indirect addressing, refer to "See also".

See also

Indirect addressing in STL (Page 737)

Indirect addressing in STL

In STL, the following options are available for indirect addressing:

- Memory-indirect addressing
- Register-indirect area-internal addressing
- Register-indirect cross-area addressing

Memory-indirect addressing

In the case of memory-indirect addressing, you store the address in a PLC variable. Double words from the operand ranges Data (DBD and DID), bit memory (MD) and local data (LD) are available for storing the address.

The following example shows applications of memory-indirect addressing:

Addressing in STL	Description
U E [MD 2]	// Execute an AND logic operation with the input bit. The address is located in the memory double word MD2.
= DIX [DBD 2]	// Assign the signal state of the RLO bit to the instance data bit. The address is located in the data double word DBD.
L EB [DID 4]	// Load the input byte in ACCU 1. The address is located in the instance double word DID4.
AUF DB [LW 2]	// Open the data block. The number of the data block is located in the local data word LW2.

Register indirect area-internal addressing

Register indirect addressing uses an address register to pick up the address of the operand. With register-indirect area-internal addressing, you program in the instruction an operand range for which the address in the address register is to apply. The address in the address register then moves in the operand range that is specified in the instruction.

The following example shows an application of register-indirect area-internal addressing:

STL	Description
LAR1 P#10.0	// Load pointer (P#10.0) in address register 1
L EW [AR1, P#2.0]	// Add content of address register 1 (P#10.0) to pointer P#2.0
	// Load content of input word IW12 into accumulator 1

Register indirect cross-area addressing

With register-indirect cross-area addressing, you program only the operand width in the instruction. The operand range and the address of the operand are indexed via the address register and can be changed in dynamic form.

The following example shows an application of register-indirect cross-area addressing:

```
LAR1 P#M10.0           // Load pointer (P#M10.0) in address register 1
L W [AR1, P#2.0]       // Add content of address register 1 (P#M10.0) to pointer
                       P#2.0
                       // Load content of memory word "MW12" into accumulator 1
LAR1 P#A10.0           // Load pointer (P#A10.0) in address register 1
L W [AR1, P#2.0]       // Add content of address register 1 (P#A10.0) to pointer
                       P#2.0
                       // Load content of output word QW12.0 into accumulator 1
```

See also

Basics of indirect addressing (Page 732)

Indirect addressing in SCL (Page 735)

Addressing structured variables (Page 729)

Basic information about address registers (Page 737)

Enter constants

Description

You have the following options for using constants in the program:

- Enter the value
- Input of a symbolic name defined in the PLC tag table (with S7-1200). The symbolic name of a constant will be automatically included in quotation marks.

Both types of constant are displayed in blue in the program.

Syntax

- Enter a value:

  <Value>
- Input of a symbolic constant name from the PLC tag table:

  "<Name>"

Example

The following examples show the use of constants:

Addressing	Description
4	Value entry for a constant of Integer type
FALSE	Value entry for a constant of Bool type
"Name"	Symbolic constants from the PLC variable table
"Offset"	Symbolic constants from the PLC variable table

More information

For more information on data types of constants and their input formats and value ranges, refer to the "Data types" section under "See also".

See also

Constants (Page 718)

Declaring constants (Page 835)

9.1.1.5 Data types

Overview of the valid data types

Validity of data type groups

The data type groups define the properties of the data, for example, the representation of the content and the valid memory areas. In the user program, you can use predefined data type or also data types that you have defined.

The following tables show the availability of predefined data types in the various S7-CPUs:

Table 9- 1 Binary numbers

Binary numbers	S7-300/400	S7-1200
BOOL (Page 745)	X	X
Bit strings		
BYTE (Page 746)	X	X
WORD (Page 747)	X	X
DWORD (Page 748)	X	X

Table 9- 2 Integers

Integers	S7-300/400	S7-1200
SINT (Page 749)	-	X
INT (Page 751)	X	X
DINT (Page 753)	X	X
USINT (Page 750)	-	X
UINT (Page 752)	-	X
UDINT (Page 754)	-	X

Table 9- 3 Floating-point numbers

Floating-point numbers	S7-300/400	S7-1200
REAL (Page 755)	X	X
LREAL (Page 756)	-	X

Table 9- 4 Timers

Timers	S7-300/400	S7-1200
TIME (Page 757)	X	X
S5TIME (Page 758)	X	-

Table 9- 5 Date and time

Date and time	S7-300/400	S7-1200
DATE (Page 759)	X	X
TIME_OF_DAY (TOD) (Page 760)	X	X
DT (DATE_AND_TIME) (Page 761)	X	-
DTL (Page 762)	-	X

Table 9- 6 Character

Character	S7-300/400	S7-1200
CHAR (Page 763)	X	X
STRING (Page 763)	X	X

Table 9- 7 Array

Array	S7-300/400	S7-1200
ARRAY [….] OF <type> (Page 765)	X	X

Table 9- 8 Structures

Structures	S7-300/400	S7-1200
STRUCT (Page 768)	X	X

Table 9- 9 Pointer

Pointer	S7-300/400	S7-1200
POINTER (Page 769)	X	-
ANY (Page 771)	X	-
VARIANT (Page 773)	-	X

Table 9- 10 Parameter types

Parameter types	S7-300/400	S7-1200
TIMER (Page 774)	X	-
COUNTER (Page 774)	X	-
BLOCK_FC (Page 774)	X	-
BLOCK_FB (Page 774)	X	-
BLOCK_DB (Page 774)	X	-
BLOCK_SDB (Page 774)	X	-
BLOCK_SFB (Page 774)	X	-

Parameter types	S7-300/400	S7-1200
BLOCK_SFC (Page 774)	X	-
VOID (Page 774)	X	X

Table 9- 11 PLC data types

PLC data types	S7-300/400	S7-1200
PLC data type (Page 775)	X	X

Table 9- 12 System data types

System data types	S7-300/400	S7-1200
IEC_TIMER (Page 776)	X[1)]	X
IEC_SCOUNTER (Page 776)	-	X
IEC_USCOUNTER (Page 776)	-	X
IEC_COUNTER (Page 776)	X[2)]	X
IEC_UCOUNTER (Page 776)	-	X
IEC_DCOUNTER (Page 776)	-	X
IEC_UDCOUNTER (Page 776)	-	X
ERROR_STRUCT (Page 776)	-	X
NREF (Page 776)	-	X
CREF (Page 776)	-	X
FBTREF (Page 776)	-	X
VREF (Page 776)	-	X
STARTINFO (Page 776)	X	-
SSL_HEADER (Page 776)	X	-
CONDITIONS (Page 776)	-	X
TADDR_Param (Page 776)	-	X
TCON_Param	-	X
[1)] For S7-300/400 CPUs, the data type is represented by TP, TON and TOF. [2)] For S7-300/400 CPUs, the data type is represented by CTU, CTD and CTUD.		

Table 9- 13 Hardware data types

Hardware data types	S7-300/400	S7-1200
REMOTE (Page 778)	-	X
HW_ANY (Page 778)	-	X
HW_IO (Page 778)	-	X
HW_SUBMODULE (Page 778)	-	X
HW_INTERFACE (Page 778)	-	X
HW_HSC (Page 778)	-	X
HW_PWM (Page 778)	-	X

Hardware data types	S7-300/400	S7-1200
HW_PTO (Page 778)	-	X
AOM_IDENT (Page 778)	-	X
EVENT_ANY (Page 778)	-	X
EVENT_ATT (Page 778)	-	X
EVENT_HWINT (Page 778)	-	X
OB_ANY (Page 778)	-	X
OB_DELAY (Page 778)	-	X
OB_TOD (Page 778)	-	X
OB_CYCLIC (Page 778)	-	X
OB_ATT (Page 778)	-	X
OB_PCYCLE (Page 778)	-	X
OB_HWINT (Page 778)	-	X
OB_COMM (Page 778)	-	X
OB_DIAG (Page 778)	-	X
OB_TIMEERROR (Page 778)	-	X
OB_STARTUP (Page 778)	-	X
PORT (Page 778)	-	X
RTM (Page 778)	-	X
CONN_ANY (Page 778)	-	X
CONN_PRG (Page 778)	-	X
CONN_OUC (Page 778)	-	X

Note

Depending on the CPU version, the actually valid data types can deviate slightly from the table.

Binary numbers

BOOL (bit)

Description

An operand of the data type BOOL represents a bit value and can contain one of the following values:

- TRUE
- FALSE

The following table shows the properties of the data type BOOL:

Length (bits)	Format	Value range	Examples of value input
1	Boolean	FALSE or TRUE	TRUE
	Binary numbers	0 or 1	1
	Octal numbers	8#0 or 8#1	8#1
	Hexadecimal numbers	16#0 or 16#1	16#1

See also

Overview of the valid data types (Page 741)

Bit strings

BYTE (byte)

Description

An operand of the data type BYTE is a bit string of 8 bits.

The following table shows the properties of the data type BYTE:

Length (bits)	Format	Value range	Examples of value input
8	Unsigned integers	0 to 255	15
	Binary numbers	2#0 to 2#11111111	2#00001111
	Octal numbers	8#0 to 8#377	8#17
	Hexadecimal numbers	B#16#0 to B#16#FF	B#16#F, 16#F

See also

Overview of the valid data types (Page 741)

Implicit conversion of BYTE (Page 784)

Explicit conversion of BYTE (Page 800)

Overview of data type conversion (Page 780)

WORD

Description

An operand of the data type WORD is a bit string of 16 bits.

The following table shows the properties of the data type WORD:

Length (bits)	Format	Value range	Examples of value input
16	Unsigned integers	0 to 65535	61680
	Binary numbers	2#0 to 2#1111111111111111	2#1111000011110000
	Octal numbers	8#0 to 8#177777	8#170360
	Hexadecimal numbers	W#16#0 to W#16#FFFF 16#0 to 16#FFFF	W#16#F0F0, 16#F0F0

See also

Overview of the valid data types (Page 741)

Implicit conversion of WORD (Page 785)

Overview of data type conversion (Page 780)

Explicit conversion of WORD (Page 801)

DWORD

Description

An operand of the data type DWORD is a bit string of 32 bits.

The following table shows the properties of the data type DWORD:

Length (bits)	Format	Value range	Examples of value input
32	Unsigned integers	0 to 4294967295	15793935
	Binary numbers	2#0 to 2#11111111111111111111111111111111	2#111100001111111100001111
	Octal numbers	8#0 to 8#37777777777	8#74177417
	Hexadecimal numbers	DW#16#0000_0000 to DW#16#FFFF_FFFF, 16#0000_0000 to 16#FFFF_FFFF	DW#16#F0FF0F, 16#F0FF0F

See also

Overview of the valid data types (Page 741)

Implicit conversion of DWORD (Page 786)

Overview of data type conversion (Page 780)

Explicit conversion of DWORD (Page 802)

Integers

SINT (8-bit integers)

Description

An operand of the data type SINT has a length of 8 bits and consists of two components: One sign and one numerical value in the two's complement. The signal stats of the bits 0 to 6 stand for the value of the number. The signal state of bit 7 represents the sign. The sign assume "0" for positive or "1" for negative.

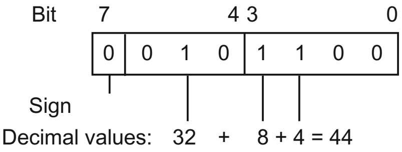
An operand of the data type SINT occupies one byte in the memory.

The following table shows the properties of the data type SINT:

Length (bits)	Format	Value range	Examples of value input
8	Signed integers	-128 to 127	+50
	Binary numbers (only positive)	2#0 to 01111111	2#01010000
	Octal numbers (only positive)	8#0 to 8#177	8#120
	Hexadecimal numbers (only positive)	16#0 to 16#7F	16#50

Example

The following figure shows the integer +44 as a binary number:



See also

Overview of the valid data types (Page 741)

Implicit conversion of SINT (Page 787)

Explicit conversion of SINT (Page 803)

USINT (8-bit integers)

Description

An operand of the data type USINT (unsigned short INT) has a length of 8 bits and can contain unsigned numerical values.

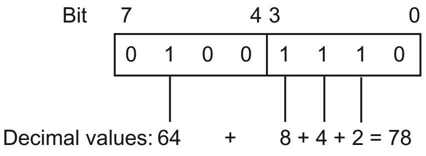
An operand of the data type USINT occupies one byte in the memory.

The following table shows the properties of the data type USINT:

Length (bits)	Format	Value range	Examples of value input
8	Unsigned integers	0 to 255	78
	Binary numbers	2#0 to 2#11111111	2#01001110
	Octal numbers	8#0 to 8#377	8#116
	Hexadecimal numbers	16#0 to 16#FF	16#4E

Example

The following figure shows the integer 78 as a binary number:



See also

Overview of the valid data types (Page 741)

Implicit conversion of USINT (Page 788)

Explicit conversion of USINT (Page 804)

INT (16-bit integers)

Description

An operand of the data type INT has a length of 16 bits and consists of two components: One sign and one numerical value in the two's complement. The signal stats of the bits 0 to 14 stand for the value of the number. The signal state of bit 15 represents the sign. The sign assume the value "0" (positive) or "1" (negative).

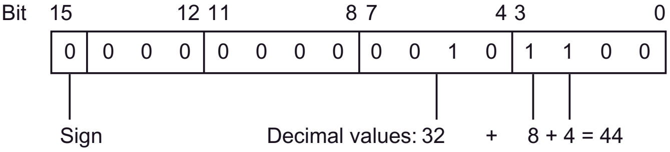
An operand of the data type INT occupies two bytes in the memory.

The following table shows the properties of the data type INT:

Length (bits)	Format	Value range	Examples of value input
16	Signed integers	- 32768 to 32767	+44
	Binary numbers (only positive)	2#0 to 2#0111111111111111	2#0000000000101100
	Octal numbers (only positive)	8#0 to 8#77777	8#54
	Hexadecimal numbers (only positive)	16#0 to 16#7FFF	16#2C

Example

The following figure shows the integer +44 as a binary number:



See also

Overview of the valid data types (Page 741)

Implicit conversion of INT (Page 789)

Explicit conversion of INT (Page 805)

UINT (16-bit integers)

Description

An operand of the data type UINT (unsigned INT) has a length of 16 bits and can contain unsigned numerical values.

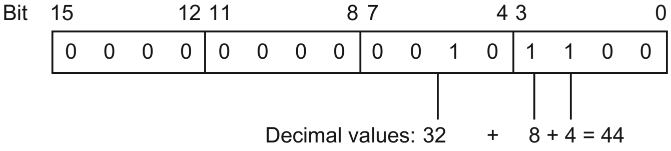
An operand of the data type UINT occupies two bytes in the memory.

The following table shows the properties of a UINT variable:

Length (bits)	Format	Value range	Examples of value input
16	Unsigned integers	0 to 65535	65295
	Binary numbers	2#0 to 2#1111111111111111	2#1111111100001111
	Octal numbers	8#0 to 8#177777	8#177417
	Hexadecimal numbers	16#0 to 16#FFFF	16# FF0F

Example

The following figure shows the integer 44 as a binary number:



See also

Overview of the valid data types (Page 741)

Implicit conversion of UINT (Page 790)

Explicit conversion of UINT (Page 807)

DINT (32-bit integers)

Description

An operand of the data type DINT has a length of 32 bits and consists of two components: One sign and one numerical value in the two's complement. The signal stats of the bits 0 to 30 stand for the value of the number. The signal state of bit 31 represents the sign. The sign assume the value "0" (positive) or "1" (negative).

An operand of the data type DINT occupies four bytes in the memory.

The following table shows the properties of the data type DINT:

Length (bits)	Format	Value range	Examples of value input
32	Signed integers	- 2 147 483 648 to + 2 147 483 647	2131754992
	Binary numbers (only positive)	2#0 to 2#0111111111111111111111 1111111111	2#0111111110000111111111 1110000
	Octal numbers (only positive)	8#0 to 8#17777777777	8#17703777760
	Hexadecimal numbers (only positive)	16#0 to 16#7FFF FFFF	16#7F0FFFF0

See also

Overview of the valid data types (Page 741)

Implicit conversion of DINT (Page 791)

Explicit conversion of DINT (Page 808)

UDINT (32-bit integers)

Description

An operand of the data type UDINT (unsigned double INT) has a length of 32 bits and can contain unsigned numerical values.

An operand of the data type DINT occupies four bytes in the memory.

The following table shows the properties of the data type UDINT:

Length (bits)	Format	Value range	Examples of value input
32	Unsigned integers	0 to 4294967295	4042322160
	Binary numbers	2#0 to 2#111111111111111111 11111111111111	2#1111000011110000111100001 1110000
	Octal numbers	8#0 to 8# 37777777777	8#36074170360
	Hexadecimal numbers	16#0000_0000 to 16# FFFF_FFFF	16#F0F0F0F0

See also

Overview of the valid data types (Page 741)

Implicit conversion of UDINT (Page 792)

Explicit conversion of UDINT (Page 809)

Floating-point numbers

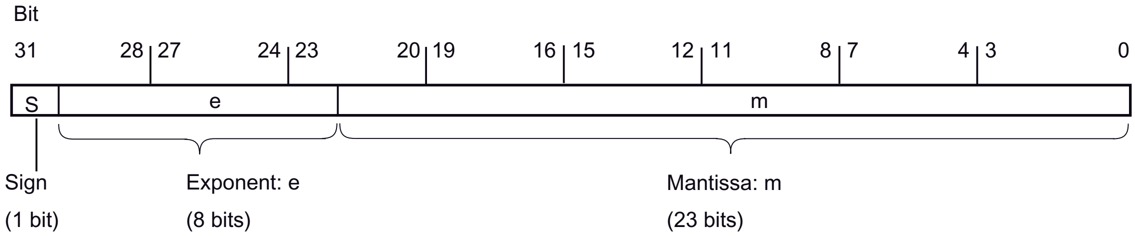
REAL

Description

Operands of the data type REAL have a length of 32 bits and are used to display floating-point numbers. An operand of the REAL data type consists of the following three components:

- Sign: The sign is determined by the signal state of bit 31. The bit 31 assume the value "0" (positive) or "1" (negative).
- 8-bit exponents to basis 2: The exponent is increased by a constant (base, +127), so that it has a value range of 0 to 255.
- 23-bit mantissa: Only the fraction part of the mantissa is shown. The integer part of the mantissa is always 1 with normalized floating-point numbers and is not stored.

The following figure shows the structure of the REAL data type:



The following table shows the properties of the REAL data type:

Length (bits)	Format	Value range	Examples of value input
32	Floating-point numbers to IEEE 754 standard	-3.402823e+38 to -1.175 495e-38 ±0 +1.175 495e-38 to +3.402823e+38	1.0e-5
	Floating-point numbers		1.0

See also

Overview of the valid data types (Page 741)

Implicit conversion of REAL (Page 793)

Explicit conversion of REAL (Page 811)

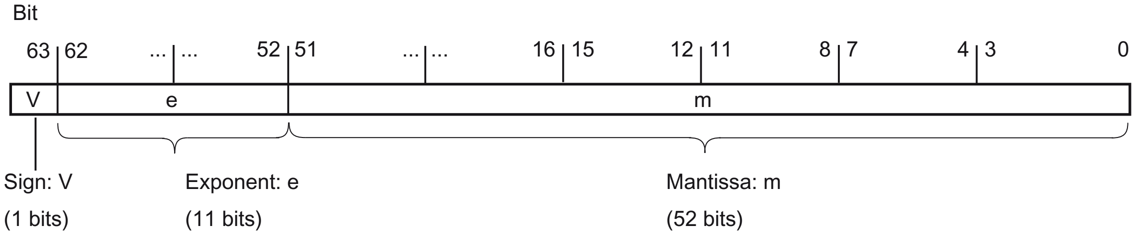
LREAL

Description

Operands of the LREAL data type have a length of 64 bits and are used to represent floating-point numbers. An operand of the LREAL data type consists of the following three components:

- Sign: The sign is determined by the signal state of bit 63. The bit 63 assumes the value "0" (positive) or "1" (negative).
- 11-bit exponents to base 2: The exponent is increased by a constant (base, +1023), so that it has a value range of 0 to 2047.
- 52-bit mantissa: Only the fraction part of the mantissa is shown. The integer part of the mantissa is always 1 with normalized floating-point numbers and is not stored.

The following figure shows the structure of the LREAL data type:



The following table shows the properties of the LREAL data type:

Length (bits)	Format	Value range	Examples of value input
64	Floating-point numbers to IEEE 754 standard	-1.7976931348623158e+308 to -2.2250738585072014e-308 ±0 +2.2250738585072014e-308 to +1.7976931348623158e+308	1.0e-5
	Floating-point numbers		1.0

See also

Overview of the valid data types (Page 741)

Explicit conversion of LREAL (Page 812)

Timers

TIME (IEC time)

Description

The contents of an operand of the data type TIME is interpreted as milliseconds. The representation contains information for days (d), hours (h), minutes (m), seconds (s) and milliseconds (ms).

The following table shows the properties of the data type TIME:

Length (bits)	Format	Value range	Examples of value input
32	Time period with sign:	T#-24d20h31m23s648ms to T#+24d20h31m23s647ms	T#10d20h30m20s630ms, TIME#10d20h30m20s630ms, 10d20h30m20s630ms

It is not necessary to specify all time units. So for instance T#5h10s is valid. If only one unit is specified, the absolute value of days, hours, and minutes must not exceed the high or low limits. When more than one time unit is specified, the value of the unit must not exceed 23 hours 59 minutes 59 seconds 999 milliseconds.

See also

Overview of the valid data types (Page 741)

Implicit conversion of TIME (Page 794)

Explicit conversion of TIME (Page 813)

S5TIME (duration)

Format

The S5TIME data type saves the duration in BCD format. The duration is the product from a time in the range 0 to 999 and a time basis. The time basis indicates the interval at which a timer decrements the time value by one unit until it reaches "0". The resolution of the times can be controlled via the time basis.

The following table shows the value range for the S5TIME data type:

Length (bits)	Format	Value range	Examples of value input
16	S7 time in increments of 10 ms (default value)	S5T#0H_0M_0S_0MS to S5T#2H_46M_30S_0MS	S5T#10s

The following table shows the time basis coding for S5TIME:

Time basis	Binary code for time basis
10 ms	00
100 ms	01
1 s	10
10 s	11

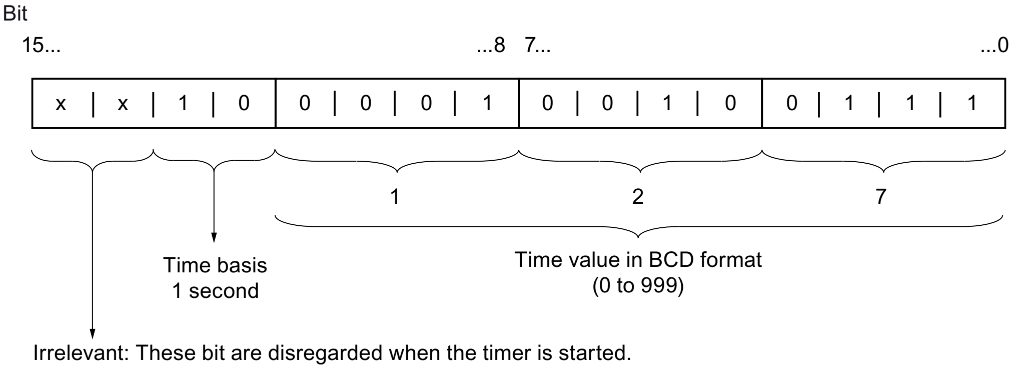
When you use the S5TIME data type with timers, you must observe limit values for the range and the resolution of time values. The table below specifies the range associated with each resolution:

Resolution	Range
0.01 s	10 ms to 9 s 990 ms
0.1 s	100 ms to 1 m 39 s 900 ms
1 s	1 s to 16 m 39 s
10 s	10 s to 2 h 46 m 30 s

Values that exceed 2h46m30s are not accepted.

Example

The following figure shows the content of the time operands for a time value of 127 and a time basis of 1 s:



See also

Overview of the valid data types (Page 741)

Date and time

DATE

Format

The DATE data type saves the date as an unsigned integer. The representation contains the year, the month, and the day.

The contents of an operand of DATE data type corresponds in hexadecimal format to the number of day since 01-01-1990 (16#0000).

The following table shows the properties of the DATE data type:

Length (bytes)	Format	Value range	Examples of value input

2	IEC date
	(Year-Month-Day)
		D#1990-1-1 to
		D#2168-12-31
			D#2009-12-31;
			DATE#2009-12-31;
			2009-12-31

	Hexadecimal numbers	W#16#0000 to W16#FF62	W#16#00F2

See also

Overview of the valid data types (Page 741)

Implicit conversion of DATE (Page 796)

Explicit conversion of DATE (Page 814)

TIME_OF_DAY (TOD)

Format

The data type TOD (TIME_OF_DAY) occupies a double word and saves the number of milliseconds since the beginning of the day (midnight) as a signless integer.

The following table shows the properties of the data type TOD:

Length (bytes)	Format	Value range	Examples of value input
4	Time-of-day (hours:minutes:seconds)	TOD#00:00:00.000 to TOD#23:59:59.999	TOD#10:20:30.400, TIME_OF_DAY#10:20:30.400
	Hexadecimal numbers	DW#16#00000000 to DW#16#05265BFF	DW16##000000F2

The specification of hours, minutes and seconds is required. The specification of milliseconds is optional.

See also

Overview of the valid data types (Page 741)

Implicit conversion of TOD (Page 795)

Explicit conversion of TOD (Page 814)

DATE_AND_TIME (date and time of day)

Format

The DT (DATE_AND_TIME) data type saves the information on date and time of day in BCD format.

The following table shows the properties of the data type DT:

Length (bytes)	Format	Value range	Example of value input
8	Date and time (Year-Month-Day-Hour:Minute:Second)	Min.: DT#1990-1-1-0:0:0 Max.: DT#2089-12-31-23:59:59.999	DT#2008-10-25-8:12:34 DATE_AND_TIME#2008-10-25-8:12:34.567

The following table shows the structure of the DT data type:

Byte	Contents	Value range
0	Year	0 to 99 (Years 1990 to 2089) BCD#90 = 1990 ... BCD#0 = 2000 ... BCD#89 = 2089
1	Month	BCD#0 to BCD#12
2	Day	BCD#1 to BCD# 31
3	Hour	BCD#0 to BCD#23
4	Minute	BCD#0 to BCD#59
5	Second[3)]	BCD#0 to BCD#59
6	The two most significant digits of MSEC	BCD#0 to BCD#999
7 (4MSB) [1)]	The least significant digit of MSEC	BCD#0 to BCD#9
7 (4LSB) [2)]	Weekday	BCD#1 to BCD#7 BCD#1 = Sunday ... BCD#7 = Saturday
[1)] MSB: Most significant bit [2)] LSB: Least significant bit [3)] Fixed point number		

See also

Overview of the valid data types (Page 741)

DTL

Description

An operand of the DTL data type has a length of 12 bytes and saves information on date and time in a predefined structure.

The following table shows the properties of the DTL data type:

Length (bytes)	Format	Value range	Example of value input
12	Date and time (Year-Month-Day:Hour:Minute:Second.Nanoseconds)	Min.: DTL#1970-01-01-00:00:00.0 Max.: DTL#2553-12-31-23:59:59.999999999	DTL#2008-12-16-20:30:20.250

The structure of the DTL data type consists of several components each of which can contain a different data type and range of values. The data type of a specified value must match the data type of the corresponding components.

The following table shows the structure components of a DTL data type and its properties:

Byte	Component	Data type	Value range
0	Year	UINT	1970 to 2553
1			
2	Month	USINT	1 to 12
3	Day	USINT	1 to 31
4	Weekday	USINT	1(Sunday) to 7(Saturday) The weekday is not considered in the value entry.
5	Hour	USINT	0 to 23
6	Minute	USINT	0 to 59
7	Second	USINT	0 to 59
8	Nanoseconds	UDINT	0 to 999999999
9			
10			
11			

See also

Overview of the valid data types (Page 741)

Explicit conversion of DTL (Page 815)

Character strings

CHAR (character)

Description

An operand of CHAR data type has a length of 8 bits and occupies one byte in the memory.

The CHAR data type stores a single character in ASCII format. You can find information on the coding of special characters at "STRING (Page 763)"

The following table shows the value range of the CHAR data type:

Length (bits)	Format	Value range	Example of value input
8	ASCII characters	ASCII character set	’A’

See also

Overview of the valid data types (Page 741)

Implicit conversion of CHAR (Page 797)

Explicit conversion of CHAR (Page 816)

STRING

Description

An operand of the STRING data type saves several characters in a character string that can consist of up to 254 characters. In a character string, all characters of the ASCII code are permitted. The characters are specified in single quotation marks.

The following table shows the properties of a STRING tag:

Length (bytes)	Format	Value range	Example of value input
n + 2 *	ASCII character string incl. special characters	0 to 254 characters	'Name'
* An operand of the STRING data type occupies two bytes more than the specified maximum length in the memory.			

A character string can also contain special characters. The alignment symbol $ is used to identify control characters, dollar character and single quotation marks.

The following table shows examples for the notation of special characters:

Character	Hex	Meaning	Example
$L or $l	0A	Line feed	'$LText', '$0AText'
$N or $n	0A and 0D	Line break The line break occupies 2 characters in the character string.	'$NText', '$0A$0DText'
$P or $p	0C	Page feed	'$PText', '$0CText'
$R or $r	0D	Carriage return (CR)	'$RText','$0DText'
$T or $t	09	Tabulator	'$TText', '$09Text'
$$	24	Dollar character	'100$$t', '100$26'
$'	27	Single quotation marks	'$'Text$'','$27Text$27'

The maximum length of the character string can be specified during the declaration of an operand using square brackets after the keyword STRING (for example, STRING[4]). If the information on maximum length is omitted, the standard length of 254 characters is set for the respective operand.

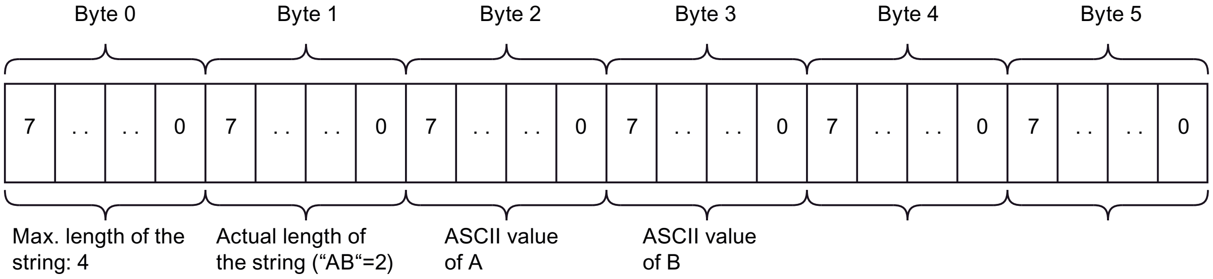
If the actual length of a specified character string is shorter than the declared maximum length, the remaining character spaces remain undefined. Only occupied character spaces are considered in the value processing.

Note

For S7-300/400 CPUs, please note: If a temporary tag of the STRING data type was defined, you must describe the BYTE "Max. length of string" with the defined length before you use the tags in the user program.

Example

The example below shows the byte sequence if the STRING[4] data type is specified with output value 'AB':



See also

Overview of the valid data types (Page 741)

Explicit conversion of STRING (Page 817)

Array

Format of ARRAY

Description

The ARRAY data type represents a field that consists of a fixed number of components of the same data type. ARRAYs cannot be nested.

The field components are addressed by means of an index. The index limits are defined in square brackets during the declaration of the field after the keyword ARRAY. The low limit value must be smaller than or equal to the high limit value. The index value must be entered directly; tags cannot be entered. A field can contain up to six dimensions, the limits of which are specified in each case separated by a comma.

The following table shows the properties of the ARRAY data type:

Length	Format	Index limits
Number of components * length of the data type	ARRAY [low limit value...high limit value] of <data type>	[-32768..32767] of <data type>

Example

The following example shows how operands of the ARRAY data type can be declared:

Name	Declaration	Comment
Measured value	ARRAY[1..20] of REAL	One-dimensional field with 20 components
Time-of-day	ARRAY[-5..5] of INT	One-dimensional field with 11 components
Character	ARRAY[1..2, 3..4] of CHAR	Two-dimensional field with 4 components

See also

Overview of the valid data types (Page 741)

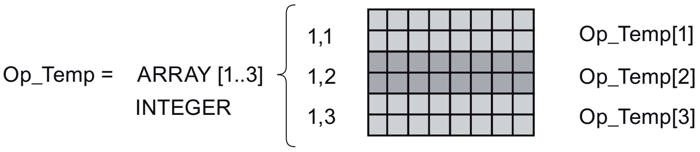
Example of a one-dimensional ARRAY

Declaration

The following table shows the declaration of a one-dimensional ARRAY variable:

Name	Data type	Comment
Op_Temp	ARRAY[1..3] of INT	One-dimensional ARRAY variable with 3 components.

The following figure shows the structure of the declared ARRAY variable:



Access to ARRAY components

The individual array components are accessed via an index.. The index of the first ARRAY component is [1], of the second [2], and of the third [3]. To access the value of the second ARRAY component, the specification "OP_Temp[2]" is required in the program in this case.

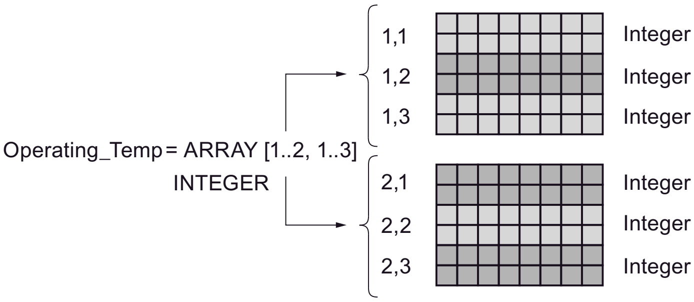
Example of a multi-dimensional ARRAY

Declaration

The following table shows the declaration of a two-dimensional ARRAY variable:

Name	Data type	Value	Comment
Betr_Temp	ARRAY[1..2,1..3] of INT	1,1,4(0)	Two-dimensional ARRAY variable with 6 components. The first two components are assigned the value "1". The remaining four components are assigned the value "0".

The following figure shows the structure of the declared ARRAY variable:



Access to the field components

The values of the individual field components are accessed via an index. The index of the first field component is, for example, [1,1] and the index of the fourth field component is [2,1]. You will, for example, have to enter "Betr_Temp[2,1]" in the program to access the value of the fourth field component.

Alternative access option

You can also declare the "Betr_Temp" variable as six-dimensional field. The following table shows an example of the declaration of a six-dimensional ARRAY variable:

Name	Data type	Value	Comment
Betr_Temp	ARRAY[1..3,1..2,1..3,1..4,1..3,1..4] of INT	-	Six-dimensional ARRAY variable

The index of the first field component is in this case [1,1,1,1,1,1] and the index of the last component is [3,2,3,4,3,4]. Intermediate values are accessed by entering the corresponding value for each dimension.

Structures

STRUCT

Description

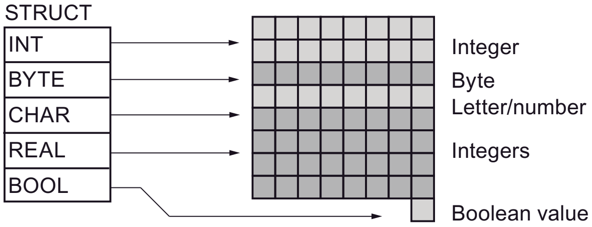
The STRUCT data type defines a structure that consists of components of different data types. Components of STRUCT or ARRAY data type can also be nested in a structure. The nesting depth is hereby limited to eight levels.

The following table shows the properties of the STRUCT data type:

Length	Format	Value range	Example of value input
A STRUCT variable starts with one byte with even address and occupies the memory up to the next word limit.	STRUCT	The value ranges of the used data types apply.	The value input rules of the used data types apply.

Example

The following figure shows an example of the structure of a STRUCT variable:



See also

Overview of the valid data types (Page 741)

Pointer

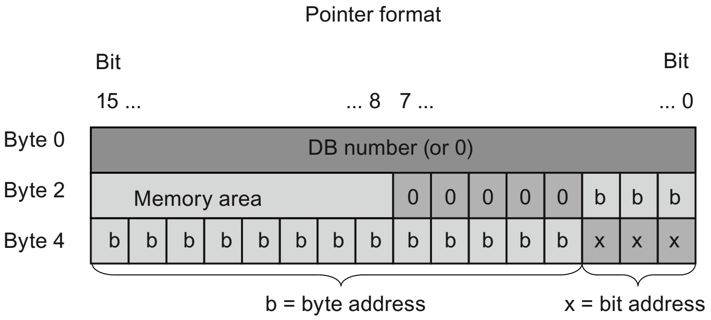
POINTER

Description

A parameter of the POINTER type is a pointer that can point to a specific variable. It occupies 6 bytes (48 bits) in the memory and can contain the following information about a variable.

- DB number, or 0 if the data is not stored in a DB
- Memory area in the CPU
- Variable address

The following figure shows the structure of the POINTER parameter type:



Types of pointer

Depending on the information, you can declare the following three types of pointer with the POINTER parameter type:

- Area-internal pointers:
  Area-internal pointers contain information on the address of a variable.
- Cross-area pointers:
  Cross-area pointers contain information on the memory area and the address of a variable.
- DB pointer:
  You can use a DB pointer to point to a data block variable. A DB pointer also contains a data block number in addition to the memory area and the address of a variable.

The following table shows the formats for the declaration of various pointer types:

Type	Format	Example of value input
Area-internal pointer	P#Byte.Bit	P#20.0
Cross-area pointer	P#OperandAreaByte.Bit	P#M20.0
DB pointer	P#Data_block.Data_operand	P#DB10.DBX20.0

You can enter a parameter of the POINTER type without prefix (P#). You entry is then automatically converted to the pointer format.

Memory areas

The following table shows the hexadecimal codes of the memory areas for the POINTER parameter type:

Hexadecimal code	Memory area	Description
B#16#80	P	I/Os
B#16#81:	E	Memory area of inputs
B#16#82	A	Memory area of outputs
B#16#83	M	Memory area of bit memory
B#16#84	DBX	Data block
B#16#85	DIX	Instance data block
B#16#86	L	Local data
B#16#87	V	Previous local data

See also

Basics of indirect addressing (Page 732)

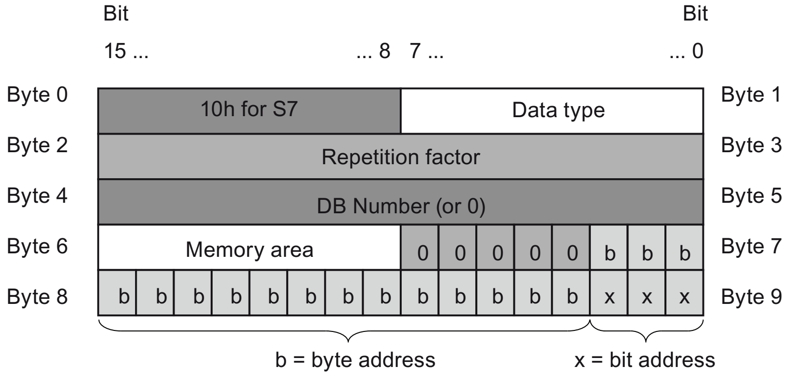
ANY

Description

A parameter of the ANY type is a pointer that points to the start of a data range and states its length. An ANY pointer occupies 10 bytes in the memory and can contain the following information:

- Data type
  Data type of the elements of the data range
- Repeat factor:
  Number of the elements of the data range
- DB number:
  Data block in which the elements of the data range are declared.
- Memory area:
  Memory area of the CPU in which the elements of the data range are stored.
- Start address of data in format "Byte.Bit"
  Start of the data range to which the ANY pointer points.

The following figure shows the structure of the ANY pointer:



An ANY pointer can recognize no structures. It can only be assigned to local variables.

The following table shows the formats for the declaration of an ANY pointer:

Format	Example of value input	Description
P#DataBlock.MemoryArea DataAddress Type Number	P#DB 11.DBX 20.0 INT 10	Area with 10 words in the global DB 11 starting with DBB 20.0
P#MemoryArea DataAddress Type Number	P#M 20.0 BYTE 10	Area with 10 bytes starting with MB 20.0
	P#E 1.0 BOOL 1	Input I 1.0

Coding of data types

The following table shows the coding of the data types for the ANY pointer:

Hexadecimal code	Data type	Description
B#16#00	NIL	Null pointer
B#16#01	BOOL	Bits
B#16#02	BYTE	bytes, 8 bits
B#16#03	CHAR	8-bit characters
B#16#04	WORD	16-bit words
B#16#05	INT	16-bit integers
B#16#37	SINT	8-bit integers
B#16#35	UINT	16-bit unsigned integers
B#16#34	USINT	8-bit unsigned integers
B#16#06	DWORD	32-bit words
B#16#07	DINT	32-bit integers
B#16#36	UDINT	32-bit unsigned integers
B#16#08	REAL	32-bit floating-point numbers
B#16#0B	TIME	Duration
B#16#0C	S5TIME	Duration
B#16#09	DATE	Date
B#16#0A	TOD	Date and time
B#16#0E	DT	Date and time
B#16#13	STRING	Character string
B#16#17	BLOCK_FB	Function block
B#16#18	BLOCK_FC	Function
B#16#19	BLOCK_DB	Data block
B#16#1A	BLOCK_SDB	System data block
B#16#1C	COUNTER	Counter
B#16#1D	TIMER	Time

Coding of the memory area

The following table shows the coding of the memory areas for the ANY pointer:

Hexadecimal code	Area	Description
B#16#80	P	I/Os
B#16#81:	I	Memory area of inputs
B#16#82	O	Memory area of outputs
B#16#83	M	Memory area of bit memory
B#16#84	DBX	Data block
B#16#85	DIX	Instance data block
B#16#86	L	Local data
B#16#87	V	Previous local data

See also

Basics of indirect addressing (Page 732)

VARIANT

Description

A parameter of the VARIANT type is a pointer that can point to variables of various data or parameter types. The VARIANT pointer can recognize structures and point to individual structure components. An operand of the VARIANT data type occupies no space in the memory.

The following table shows the properties of the VARIANT pointer:

Length (bytes)	Representation	Format	Example of value input
0	Symbolic	Operand	MyTag
		NameDataBlock.NameOperand.Component	MyDB.StructVariable.FirstComponent
	Absolute	Operand	%MW10
		DataBlockNumber.Operand Type Length (valid only for blocks with standard access)	P#DB10.DBX10.0 INT 12

See also

Basics of indirect addressing (Page 732)

Parameter types

Parameter types

Description

The parameter types are data types for formal parameters that are transferred to the called block. A parameter type can also be a PLC data type.

The following table shows the available parameter data types and their purpose:

Parameter type	Length (bits)	Description
TIMER	16	Is used to specify a timer that is used in the called logic block. If you assign a formal parameter of the TIMER parameter type, the associated actual parameter must be a timer. Example: T1
COUNTER	16	Is used to specify a counter that is used in the called logic block. If you assign a formal parameter of the COUNTER parameter type, the associated actual parameter must be a counter. Example: C10
BLOCK_FC	16	Is used to specify a block that is used as input in the called logic block. The declaration of the parameter determines the block type (for example FB, FC, DB) that is to be used. If you assign a formal parameter of the BLOCK parameter type, enter a block address as the actual parameter. Example: DB 3
BLOCK_FB	16	
BLOCK_DB	16	
BLOCK_SDB	16	
BLOCK_SFB	16	
BLOCK_SFC	16	
BLOCK_OB	16	
BLOCK_SDT	-	
BLOCK_UDT	-	
VOID	-	The VOID parameter type saves no values. This parameter type is used if the return values of an output are not required. The VOID parameter type, for example, can be specified at the STATUS output if no error information is required.

See also

Overview of the valid data types (Page 741)

Basics of PLC data types (Page 1036)

PLC data types

PLC data types

Description

PLC data types are data structures that you define and that can be used multiple times within the program. The structure of a PLC is made up of several components, each of which can contain different data types. You define the type of components during the declaration of the PLC data type.

PLC data types can be used for the following applications:

- PLC data types can be used as data types for variables in the variable declaration of logic blocks or in data blocks.
- PLC data types can be used as templates for the creation of global data blocks with identical data structures.

See also

Addressing structured variables (Page 729)

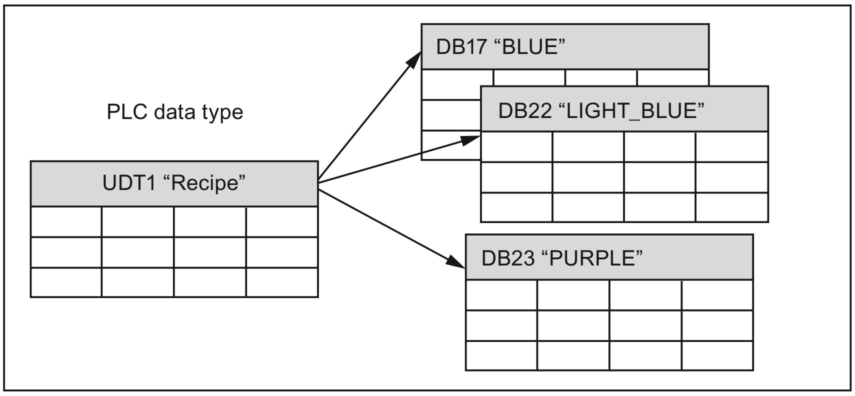
Example of a PLC data type

Example

You can declare PLC data types as the type when creating data blocks. Based on this type you can create a number of data blocks, all of which have the same data structure. These data blocks can be adapted by entering different actual values for the corresponding task.

For instance, create a PLC data type for a recipe for blending paints. You can then assign this data type to several data blocks, each of which contains other quantity information.

The following figure shows this application:



System data types

System data types

Description

The system data types (SDT) are made available by the system and have a predefined structure. The structure of a system data type consists of a fixed number of components that can have various data types. It is not possible to change the structure of a system data type.

The system data types can only be used for specific instructions. The following table shows the available system data types and their purpose:

System data type	Length (bytes)	Description
IEC_TIMER	16	Structure of a timer This data type is used for the "TP", "TOF", "TON" and "TONR" instructions, for example.
IEC_SCOUNTER	3	Structure of a counter whose count values are of SINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_USCOUNTER	3	Structure of a counter whose count values are of USINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_COUNTER	6	Structure of a counter whose count values are of INT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_UCOUNTER	6	Structure of a counter whose count values are of UINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_DCOUNTER	12	Structure of a counter whose count values are of DINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_UDCOUNTER	12	Structure of a counter whose count values are of UDINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
ERROR_STRUCT	28	Structure of an error information to a programming or I/0 access error. This data type is used, for example, for the "GET_ERROR" instruction.
CREF	8	Components of the ERROR_STRUCT data type, in which information about the address of a block is saved.

System data type	Length (bytes)	Description
NREF	8	Components of the ERROR_STRUCT data type, in which information about the address of an operand is saved.
VREF	12	Is used for storage of a VARIANT pointer. This data type is, for example, used for instructions from S7-1200 Motion Control.
STARTINFO	12	Specifies the data structure in which the start information is saved. This data type is used, for example, for the "RD_SINFO" instruction.
SSL_HEADER	4	Specifies the data structure in which information about the data records are saved during the reading of the system status lists. This data type is used, for example, for the "RDSYSST" instruction.
CONDITIONS	52	User-defined data structure defining the conditions for start and end of a data reception. This data type is used, for example, for the "RCV_CFG" instruction.
TADDR_Param	8	Specifies the structure of a data block which stores descriptions of connections for Open User Communication via UDP. This data type is used for the "TUSEND" and "TURSV" instructions, for example.
TCON_Param	64	Specifies the structure of a data block which stores descriptions of connections for Open User Communication via Industrial Ethernet (PROFINET). This data type is used for the "TSEND" and "TRSV" instructions, for example.

See also

Overview of the valid data types (Page 741)

Hardware data types

Hardware data types

Description

The hardware data types are made available by the CPU. The number of available hardware data types depends on the CPU.

Constants of a specific hardware data type are stored based on the modules set in the hardware configuration. When an instruction for controlling or activating a configured module is inserted in the user program, the available constants can be used for the parameters.

The following table shows the available hardware data types and their purpose:

Data type	Basic data type	Description
REMOTE	ANY	Is used to specify the address of a remote CPU This data type is used for the "PUT" and "GET" instructions, for example.
HW_ANY	WORD	Identification of any hardware component, such as a module
HW_IO	HW_ANY	Identification number of the CPU or the interface The number is automatically allocated and is stored in the properties of the CPU or of the interface in the hardware configuration
HW_SUBMODULE	HW_IO	Identification of a central hardware component
HW_INTERFACE	HW_SUBMODULE	Identification of an interface component
HW_HSC	HW_SUBMODULE	Identification of a fast counter This data type is used, for example, for the "CTRL_HSC" instruction.
HW_PWM	HW_SUBMODULE	Identification of a pulse width modulation This data type is used, for example, for the "CTRL_PWM" instruction.
HW_PTO	HW_SUBMODULE	Identification of a pulse encoder This data type is used for motion control
AOM_IDENT	DWORD	Identification of an object in the runtime system of the AS.
EVENT_ANY	AOM_IDENT	Used to identify any event
EVENT_ATT	EVENT_ANY	Is used to specify an event that can be assigned dynamically to an OB This data type is used for the "ATTACH" and "DETACH" instructions, for example.
EVENT_HWINT	EVENT_ATT	Is used to specify a hardware interrupt event.
OB_ANY	INT	Is used to specify any organization block
OB_DELAY	OB_ANY	Used to specify an organization block that is called when a time-delay interrupt occurs. This data type is used, for example, for the "SRT_DINT" and "CAN_DINT" instructions.

Data type	Basic data type	Description
OB_TOD	OB_ANY	Specifies the number of a time-of-day interrupt organization block This data type is used, for example, for the "SET_TINT", "CAN_TINT", "ACT_TINT" and "QRY_TINT" instructions.
OB_CYCLIC	OB_ANY	Is used to specify an organization block that is called when a watchdog interrupt occurs.
OB_ATT	OB_ANY	Is used to specify an organization block that can be assigned dynamically to an event. This data type is used for the "ATTACH" and "DETACH" instructions, for example.
OB_PCYCLE	OB_ANY	Is used to specify an organization block that can be assigned to an event of the "Cyclic program" event class.
OB_HWINT	OB_ATT	Is used to specify an organization block that is called when a hardware interrupt occurs.
OB_DIAG	OB_ANY	Is used to specify an organization block that is called when a diagnostic interrupt occurs.
OB_TIMEERROR	OB_ANY	Is used to specify an organization block that is called when a time error occurs.
OB_STARTUP	OB_ANY	Is used to specify an organization block that is called when a startup event occurs.
PORT	UINT	Is used to specify a communication port This data type is used for point-to-point communication.
RTM	UINT	Is used to specify the number of an operating hours counter This data type is used, for example, for the "RTM" instruction.
CONN_ANY	WORD	Used to identify any connection
CONN_PRG	CONN_ANY	Is used to specify a connection for open communication via UDP
CONN_OUC	CONN_ANY	Used to identify a connection for open communication via Industrial Ethernet (PROFINET)

See also

Overview of the valid data types (Page 741)

Data type conversion

Data type conversion

Overview of data type conversion

Introduction

If you link several operands in an instruction, you must make sure that the data types are compatible. This applies also for assignments or for supplying block parameters. If the operands are not the same data type, a conversion has to be carried out.

There are two options for the conversion:

- Implicit conversion

  The conversion take place automatically when the instruction is executed.

- Explicit conversion

  You use an explicit conversion instruction before the actual instruction is executed.

Implicit conversion

An implicit conversion is executed automatically if the data types of the operands are compatible. This compatibility test can be carried out according to criteria that are more or less strict:

- With IEC check
  If IEC check is set the following rules are applied:
  - The implicit conversion of bit strings into other data types is not possible. An operand of the WORD data type, for example, cannot be specified at a parameter if the INT data type is expected at this parameter.
  - The bit length of the source data type must not exceed the bit length of the destination data type. An operand of the WORD data type, for example, cannot be specified at a parameter if the BYTE data type is expected at this parameter.
- Without IEC check (default setting)
  If IEC check is not set the following rules are applied:
  - The implicit conversion of bit strings into other data types is possible. An operand of the WORD data type, for example, can be specified at a parameter if the INT data type is expected at this parameter.
  - The implicit conversion of bit strings into floating-point numbers is not possible. An operand of the WORD data type, for example, cannot be specified at a parameter if the REAL data type is expected at this parameter.
  - The implicit conversion of bit strings into the TIME, TOD, DATE and CHAR data types is only possible if the bit length is identical. An operand of the WORD data type, for example, cannot be specified at a parameter if the DATE data type is expected at this parameter.
  - The bit length of the source data type must not exceed the bit length of the destination data type. An operand of the DINT data type, for example, cannot be specified at a parameter if the INT data type is expected at this parameter.
  - At in/out parameters, the bit length of a specified operand must match the programmed bit length of the respective parameter.

In the implicit conversion of data types with a smaller bit length to data types with greater bit length (for example, from BYTE to INT) the parameters of the conversion boxes are identified with a gray symbol.

For more information about the setting of the IEC check and the implicit conversion, refer to "See also".

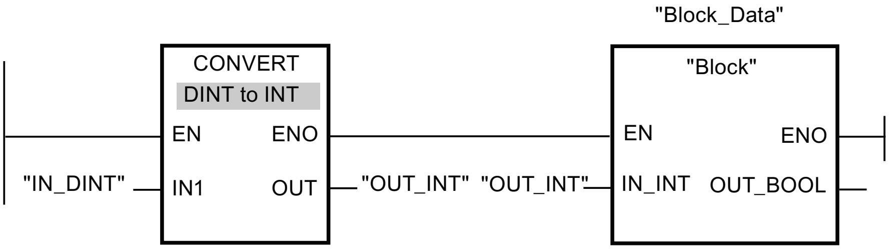
Explicit conversion

If the operands are not compatible and an implicit conversion is therefore not possible, you can use an explicit conversion instruction. You can find the conversion instructions in the "Instructions" task card.

A possible overflow is displayed at the ENO enable output. An overflow is created, for example, if the value of the source data type is greater than the value of the target data type.

For more information about explicit conversion, refer to "See also".

The following figure shows an example in which an explicit data type conversion must be carried out:



The "Block" function block expects a tag of the INT data type at the "IN_INT" input parameter. Therefore, the value of the "IN_DINT" tag must first be converted from DINT to INT. If the value of the "IN_DINT" tag is within the permitted value range of the INT data type, the conversion takes place. Otherwise, an overflow is signaled. A conversion still takes place even in case of an overflow, but the values are cut off and the enable output ENO is set to "0".

See also

Setting and canceling the IEC check (Page 782)

Implicit conversion

Setting and canceling the IEC check

When an instruction is executed, the data types are checked for compatibility to the employed operands. This compatibility test can be carried out according to criteria that are more or less strict. If "IEC Check" is activated, stricter criteria are applied.

You can set the IEC check centrally for all new blocks of the project or for specific blocks.

Setting IEC check for new blocks

To set the IEC check for all new blocks in the project, proceed as follows:

1. Select the "Settings" command in the "Options" menu.

   The "Settings" window is displayed in the work area.

2. Select the "PLC programming > General" group in the area navigation.
3. Select or clear the "IEC Check" check box in the "Default settings for new blocks" group.

   The IEC check is enabled or disabled for all new blocks in the program.

Setting IEC check for a block

To set the IEC check for a block, proceed as follows:

1. Open the block.
2. Open the "Properties" tab in the inspector window.
3. Click the "Attributes" group in area navigation.
4. Select or clear the "IEC Check" check box.

   The IEC check is enabled or disabled for this block. The setting is stored together with the project.

Binary numbers

Implicit conversion of BOOL

Options for the implicit conversion of LAD and FBD

The implicit conversion of the BOOL data type is not possible.

Options for implicit conversion of SCL

The following table shows the options for implicit conversion of the BOOL data type:

Source	Destination	With IEC Check	Without IEC Check	Explanation
BOOL	BYTE	x	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	x	x	
	DWORD	x	x	
x: Conversion possible -: Conversion not possible				

See also

BOOL (bit) (Page 745)

Bit strings

Implicit conversion of BYTE

Options for implicit conversion of LAD and FBD

The following table shows the options for implicit conversion of the BYTE data type:

Source	Destination	With IEC Check	Without IEC Check	Explanation
BYTE	BOOL	-	-	No implicit conversion
	WORD	x	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	DWORD	x	x	
	SINT	-	x	
	USINT	-	x	
	INT	-	x	
	UINT	-	x	
	DINT	-	x	
	UDINT	-	x	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	x	The bit pattern of the source value is transferred unchanged to the destination data type.
x: Conversion possible -: Conversion not possible				

See also

BYTE (byte) (Page 746)

Setting and canceling the IEC check (Page 782)

Overview of data type conversion (Page 780)

Explicit conversion of BYTE (Page 800)

Implicit conversion of WORD

Options for implicit conversion

The following table shows the options for the implicit conversion of WORD data type:

Source	Destination	With IEC Check	Without IEC Check	Description
WORD	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	DWORD	x	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	UINT	-	x	
	DINT	-	x	
	UDINT	-	x	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	TOD	-	-	
	DATE	-	x	The bit pattern of the source value is transferred unchanged to the destination data type.
	STRING	-	-	No implicit conversion
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

WORD (Page 747)

Setting and canceling the IEC check (Page 782)

Overview of data type conversion (Page 780)

Explicit conversion of WORD (Page 801)

Implicit conversion of DWORD

Options for implicit conversion

The following table shows the options for the implicit conversion of DWORD data type:

Source	Destination	With IEC Check	Without IEC Check	Description
DWORD	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	x	The bit pattern of the source value is transferred unchanged to the destination data type.
	UDINT	-	x	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	x	The bit pattern of the source value is transferred unchanged to the destination data type.
	DTL	-	-	No implicit conversion
	TOD	-	x	The bit pattern of the source value is transferred unchanged to the destination data type.
	DATE	-	-	No implicit conversion
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

DWORD (Page 748)

Setting and canceling the IEC check (Page 782)

Overview of data type conversion (Page 780)

Explicit conversion of DWORD (Page 802)

Integers

Implicit conversion of SINT

Options for implicit conversion

The following table shows the options for the implicit conversion of SINT data type:

Source	Destination	With IEC Check	Without IEC Check	Description
SINT	BOOL	-	-	No implicit conversion
	BYTE	-	x	The bit pattern of the source value is transferred unchanged to the destination data type.
	WORD	-	-	No implicit conversion
	DWORD	-	-	
	USINT	-	-	
	INT	x	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	UINT	-	-	No implicit conversion
	DINT	x	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	UDINT	-	-	No implicit conversion
	REAL	x	x	The value is converted to the format of the destination data type. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	x	x	
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

SINT (8-bit integers) (Page 749)

Setting and canceling the IEC check (Page 782)

Overview of data type conversion (Page 780)

Explicit conversion of SINT (Page 803)

Implicit conversion of USINT

Options for implicit conversion

The following table shows the options for the implicit conversion of USINT data type:

Source	Destination	With IEC Check	Without IEC Check	Description
USINT	BOOL	-	-	No implicit conversion
	BYTE	-	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	x	
	DWORD	-	x	
	SINT	-	-	No implicit conversion
	INT	x	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	UINT	x	x	
	DINT	x	x	
	UDINT	x	x	
	REAL	x	x	The value is converted to the format of the destination data type. (The value "1", for example, is converted to the value "1.0".)
	LREAL	x	x	
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

USINT (8-bit integers) (Page 750)

Setting and canceling the IEC check (Page 782)

Overview of data type conversion (Page 780)

Explicit conversion of USINT (Page 804)

Implicit conversion of INT

Options for implicit conversion

The following table shows the options for the implicit conversion of INT data type:

Source	Destination	With IEC Check	Without IEC Check	Description
INT	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	x	The bit pattern of the source value is transferred unchanged to the destination data type.
	DWORD	-	-	No implicit conversion
	SINT	-	-	
	USINT	-	-	
	UINT	-	-	
	DINT	x	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	UDINT	-	-	No implicit conversion
	REAL	x	x	The value is converted to the format of the destination data type. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	x	x	
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

INT (16-bit integers) (Page 751)

Setting and canceling the IEC check (Page 782)

Overview of data type conversion (Page 780)

Explicit conversion of INT (Page 805)

Implicit conversion of UINT

Options for implicit conversion

The following table shows the options for the implicit conversion of UINT data type:

Source	Destination	With IEC Check	Without IEC Check	Description
UINT	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	DWORD	-	x	
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	DINT	x	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	UDINT	x	x	
	REAL	x	x	The value is converted to the format of the destination data type. (The value "1", for example, is converted to the value "1.0".)
	LREAL	x	x	
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	STRING	-	-	No implicit conversion
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

UINT (16-bit integers) (Page 752)

Setting and canceling the IEC check (Page 782)

Overview of data type conversion (Page 780)

Explicit conversion of UINT (Page 807)

Implicit conversion of DINT

Options for implicit conversion

The following table shows the options for the implicit conversion of DINT data type:

Source	Destination	With IEC Check	Without IEC Check	Description
DINT	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	x	The bit pattern of the source value is transferred unchanged to the destination data type.
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	UDINT	-	-	
	REAL	-	-	
	LREAL	x	x	The value is converted to the format of the destination data type. (The value "-1", for example, is converted to the value "-1.0".)
	TIME	-	x	The bit pattern of the source value is transferred unchanged to the destination data type.
	DTL	-	-	No implicit conversion
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

DINT (32-bit integers) (Page 753)

Setting and canceling the IEC check (Page 782)

Overview of data type conversion (Page 780)

Explicit conversion of DINT (Page 808)

Implicit conversion of UDINT

Options for implicit conversion

The following table shows the options for the implicit conversion of UDINT data type:

Source	Destination	With IEC Check	Without IEC Check	Description
UDINT	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	x	The bit pattern of the source value is transferred unchanged to the destination data type.
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	REAL	-	-	
	LREAL	x	x	The value is converted to the format of the destination data type. (The value "1", for example, is converted to the value "1.0".)
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	x	The bit pattern of the source value is transferred unchanged to the destination data type.
	DATE	-	-	No implicit conversion
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

UDINT (32-bit integers) (Page 754)

Setting and canceling the IEC check (Page 782)

Overview of data type conversion (Page 780)

Explicit conversion of UDINT (Page 809)

Floating-point numbers

Implicit conversion of REAL

Options for implicit conversion

The following table shows the options for the implicit conversion of REAL data type:

Source	Destination	With IEC Check	Without IEC Check	Description
REAL	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LREAL	x	x	The value is transferred to the destination data type.
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

REAL (Page 755)

Setting and canceling the IEC check (Page 782)

Overview of data type conversion (Page 780)

Explicit conversion of REAL (Page 811)

Implicit conversion of LREAL

Options for implicit conversion

The implicit conversion of LREAL data type is not possible.

See also

Explicit conversion of LREAL (Page 812)

Timers

Implicit conversion of TIME

Options for implicit conversion

The following table shows the options for the implicit conversion of TIME data type:

Source	Destination	With IEC Check	Without IEC Check	Description
TIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	x	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in milliseconds.
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	x	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in milliseconds.
	UDINT	-	-	No implicit conversion
	REAL	-	-	
	LREAL	-	-	
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	

Source	Destination	With IEC Check	Without IEC Check	Description
x: Conversion possible -: Conversion not possible				

See also

TIME (IEC time) (Page 757)

Setting and canceling the IEC check (Page 782)

Overview of data type conversion (Page 780)

Explicit conversion of TIME (Page 813)

Date and time

Implicit conversion of DTL

Options for implicit conversion

The implicit conversion of DTL data type is not possible.

See also

Explicit conversion of DTL (Page 815)

Implicit conversion of TOD

Options for implicit conversion

The following table shows the options for the implicit conversion of TOD data type:

Source	Destination	With IEC Check	Without IEC Check	Description
TOD	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	x	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00 hrs).
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	

Source	Destination	With IEC Check	Without IEC Check	Description
	UINT	-	-	
	DINT	-	-	
	UDINT	-	x	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00 hrs).
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

TIME_OF_DAY (TOD) (Page 760)

Setting and canceling the IEC check (Page 782)

Overview of data type conversion (Page 780)

Explicit conversion of TOD (Page 814)

Implicit conversion of DATE

Options for implicit conversion

The following table shows the options for the implicit conversion of DATE data type:

Source	Destination	With IEC Check	Without IEC Check	Description
DATE	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	x	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of days since 01-01-1990.
	DWORD	-	-	No implicit conversion
	SINT	-	-	
	USINT	-	-	
	INT	-	-	

Source	Destination	With IEC Check	Without IEC Check	Description
	UINT	-	x	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of days since 01-01-1990.
	DINT	-	-	No implicit conversion
	UDINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	TOD	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

DATE (Page 759)

Setting and canceling the IEC check (Page 782)

Overview of data type conversion (Page 780)

Explicit conversion of DATE (Page 814)

Character strings

Implicit conversion of CHAR

Options for implicit conversion

The following table shows the options for the implicit conversion of CHAR data type:

Source	Destination	With IEC Check	Without IEC Check	Description
CHAR	BOOL	-	-	No implicit conversion
	BYTE	-	x	The bit pattern of the source value is transferred unchanged to the destination data type.
	WORD	-	-	No implicit conversion
	DWORD	-	-	
	SINT	-	-	
	USINT	-	-	

Source	Destination	With IEC Check	Without IEC Check	Description
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
x: Conversion possible -: Conversion not possible				

See also

CHAR (character) (Page 763)

Setting and canceling the IEC check (Page 782)

Overview of data type conversion (Page 780)

Explicit conversion of CHAR (Page 816)

Implicit conversion of STRING

Options for implicit conversion

The implicit conversion of STRING data type is not possible.

See also

Explicit conversion of STRING (Page 817)

Explicit conversion

Binary numbers

Explicit conversion of BOOL

Options for explicit conversion in LAD, FBD, STL and GRAPH

The explicit conversion of the BOOL data type is not possible.

Options for explicit conversion in SCL

The following table shows the options and instructions for the explicit conversion of the BOOL data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
BOOL	BYTE	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	CONVERT
	WORD	x		
	DWORD	x		
	INT	x		BOOL_TO_INT
	DINT	x		BOOL_TO_DINT
	REAL	-	No explicit conversion	-
	TIME	-		
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		
	STRING	-		
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

BOOL (bit) (Page 745)

Bit strings

Explicit conversion of BYTE

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BYTE data type. If the permitted value range of the destination data type is violated, the enable out ENO is set to "0". In this case the result of the conversion is invalid.

Source	Destination	Conversion	Description	Mnemonics of Instruction
BYTE[1]	BOOL[2]	x	The lowest bit is transferred into the destination data type.	CONVERT
	WORD[1]	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	
	DWORD[1]	x		
	SINT	x		
	USINT	x		
	INT	x		
	UINT	x		
	DINT	x		
	UDINT	x		
	REAL	-	No explicit conversion	-
	LREAL	-		
	TIME	-		
	DTL	-		
	TOD	-		
	DATE	-		
	STRING	-		
	CHAR	-		
x: Conversion possible - : Conversion not possible [1] Bit strings (BYTE, WORD, DWORD) are interpreted as unsigned integer with the same bit length. The BYTE data type is interpreted as USINT, WORD as UINT, and DWORD as UDINT. [2] Is valid for SCL only				

See also

BYTE (byte) (Page 746)

Implicit conversion of BYTE (Page 784)

Overview of data type conversion (Page 780)

Explicit conversion of WORD

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the WORD data type. If the permitted value range of the destination data type is violated, the enable out ENO is set to "0". In this case the result of the conversion is invalid.

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
WORD[1]	BOOL[2]	x	The lowest bit is transferred into the destination data type.	CONVERT
	BYTE[1]	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	
	DWORD[1]	x		
	SINT	x		
	USINT	x		
	INT	x		
	UINT	x		
	DINT	x		
	UDINT	x		
	REAL	-	No explicit conversion	-
	LREAL	-		
	TIME	-		
	DTL	-		
	TOD	-		
	DATE	-		
	STRING	-		
	CHAR	-		
	BLOCK_DB	x	The bit pattern of WORD is interpreted as a data block number. This conversion is only possible in data blocks with standard access.	WORD_TO_BLOCK_DB
WORD_BCD	INT	-	No explicit conversion	-
BCD	INT	-		

x: Conversion possible

- : Conversion not possible

[1] Bit strings (BYTE, WORD, DWORD) are interpreted as unsigned integer with the same bit length. The BYTE data type is interpreted as USINT, WORD as UINT, and DWORD as UDINT.

[2] Applies to SCL only

See also

WORD (Page 747)

Implicit conversion of WORD (Page 785)

Overview of data type conversion (Page 780)

Explicit conversion of DWORD

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the DWORD data type. If the permitted value range of the destination data type is violated, the enable out ENO is set to "0". In this case the result of the conversion is invalid.

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DWORD[1]	BOOL[2]	x	The lowest bit is transferred into the destination data type.	CONVERT
	BYTE[1]	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	
	WORD[1]	x		
	SINT	x		
	USINT	x		
	INT	x		
	UINT	x		
	DINT	x		
	UDINT	x		
	REAL	x[2]	The bit pattern of the source value is transferred unchanged to the destination data type.	
	LREAL	-	No explicit conversion	-
	TIME	-		
	DTL	-		
	TOD	-		
	DATE	-		
	STRING	-		
	CHAR	-		
DWORD_BCD	DINT	-		
BCD	DINT	-		

x: Conversion possible

- : Conversion not possible

[1] Bit strings (BYTE, WORD, DWORD) are interpreted as unsigned integer with the same bit length. The BYTE data type is interpreted as USINT, WORD as UINT, and DWORD as UDINT.

[2] Applies to SCL only

See also

DWORD (Page 748)

Implicit conversion of DWORD (Page 786)

Overview of data type conversion (Page 780)

Integers

Explicit conversion of SINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the SINT data type. If the permitted value range of the destination data type is violated, the enable out ENO is set to "0". In this case the result of the conversion is invalid.

Source	Destination	Conversion	Description	Mnemonics of the instruction
SINT	BOOL[2)]	x	The lowest bit is transferred into the destination data type.	CONVERT
	BYTE[1)]	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	
	WORD[1)]	x		
	DWORD[1)]	x		
	USINT	x		
	INT	x		
	UINT	x		
	DINT	x		
	UDINT	x		
	REAL	x	The value is converted into the format of the destination data type (the value "-1", for example, is converted with "Convert" instruction into the value "-1.0"). The result of the conversion depends on the mode of functioning of the instruction used.	CONVERT, NORM_X
	LREAL	x		
	TIME	-	No explicit conversion	-
	DTL	-		
	TOD	-		
	DATE	-		
	STRING	x	The value is converted into a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	S_CONV, VAL_STRG, CONVERT[2)]
	CHAR[1)]	x	The bit pattern of the source value is transferred unchanged to the destination data type. If negative values are converted, the enable output ENO is set to "0".	CONVERT

Source	Destination	Conversion	Description	Mnemonics of the instruction
x: Conversion possible - : Conversion not possible [1] Bit strings (BYTE, WORD, DWORD) and the CHAR data type are interpreted as unsigned integer with the same bit length. The BYTE data type is interpreted as USINT, WORD as UINT, and DWORD as UDINT. The CHAR data type is interpreted as USINT. [2] Is valid for SCL only				

See also

SINT (8-bit integers) (Page 749)

Implicit conversion of SINT (Page 787)

Overview of data type conversion (Page 780)

Explicit conversion of USINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of USINT data type:

Source	Destination	Conversion	Description	Mnemonics of the instruction
USINT	BOOL[2]	x	The lowest bit is transferred into the destination data type. If the permitted value range of the destination data type is violated, the enable out ENO is set to "0". In this case the result of the conversion is invalid.	CONVERT
	BYTE[1]	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	
	WORD[1]	x		
	DWORD[1]	x		
	SINT	x	The bit pattern of the source value is transferred unchanged to the destination data type. If the sign bit is overwritten during the conversion, the enable output ENO is set to "0".	
	INT	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	
	UINT	x		
	DINT	x		
	UDINT	x		
	REAL	x	The value is converted into the format of the	CONVERT, NORM_X

Source	Destination	Conversion	Description	Mnemonics of the instruction
	LREAL	x	destination data type (the value "1", for example, is converted with "Convert" instruction into the value "1.0"). The result of the conversion depends on the mode of functioning of the instruction used.	
	TIME	-	No explicit conversion	-
	DTL	-		
	TOD	-		
	DATE	-		
	STRING	x	The value is converted into a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	S_CONV, VAL_STRG, CONVERT[2)]
	CHAR[1)]	x	The bit pattern of the source value is transferred unchanged to the destination data type.	CONVERT
x: Conversion possible - : Conversion not possible [1)] Bit strings (BYTE, WORD, DWORD) and the CHAR data type are interpreted as unsigned integer with the same bit length. The BYTE data type is interpreted as USINT, WORD as UINT, and DWORD as UDINT. The CHAR data type is interpreted as USINT. [2)] Is valid for SCL only				

See also

USINT (8-bit integers) (Page 750)

Implicit conversion of USINT (Page 788)

Overview of data type conversion (Page 780)

Explicit conversion of INT

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the INT data type. If a negative value in an unsigned target data type is converted or if an overflow occurs, the enable output ENO is set to "0" and the result of the conversion is invalid.

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
INT	BOOL[2)]	x	The lowest bit is transferred into the destination data type.	CONVERT
	BYTE[1)]	x	The bit pattern of the source value is transferred unchanged right-justified to the	
	WORD[1)]	x		

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	DWORD[1]	x	destination data type.	
	SINT	x		
	USINT	x		
	UINT	x		
	DINT	x		
	UDINT	x		
	REAL	x	The value is converted into the format of the destination data type (the value "1", for example, is converted with "Convert" instruction into the value "1.0"). The result of the conversion depends on the mode of functioning of the instruction used.	CONVERT, NORM_X
	LREAL	x		
	TIME	-	No explicit conversion	-
	DTL	-		
	TOD	-		
	DATE	-		
	STRING	x	The value is converted into a character string. The character string is shown preceded by a sign. If the length of the character string is violated, the enable output ENO is set to "0".	S_CONV, VAL_STRG, CONVERT[2]
	CHAR[1]	x	The bit pattern of the source value is transferred unchanged to the destination data type.	CONVERT
	BCD	-	No explicit conversion	-
	BCD_WORD	-		

x: Conversion possible

- : Conversion not possible

[1] Bit strings (BYTE, WORD, DWORD) and the CHAR data type are interpreted as unsigned integer with the same bit length. The BYTE data type is interpreted as USINT, WORD as UINT, and DWORD as UDINT. The CHAR data type is interpreted as USINT.

[2] Applies to SCL only

See also

INT (16-bit integers) (Page 751)

Implicit conversion of INT (Page 789)

Overview of data type conversion (Page 780)

Explicit conversion of UINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the UINT data type. If the permitted value range of the destination data type is violated, the enable out ENO is set to "0". In this case the result of the conversion is invalid.

Source	Destination	Conversion	Description	Mnemonics of the instruction
UINT	BOOL2)	x	The lowest bit is transferred into the destination data type.	CONVERT
	BYTE1)	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	
	WORD1)	x		
	DWORD1)	x		
	SINT	x		
	USINT	x		
	INT	x	The bit pattern of the source value is transferred unchanged to the destination data type. If the sign bit is overwritten during the conversion, the enable output ENO is set to "0".	
	DINT	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	
	UDINT	x		
	REAL	x	The value is converted into the format of the destination data type (the value "1", for example, is converted with "Convert" instruction into the value "1.0"). The result of the conversion depends on the mode of functioning of the instruction used.	CONVERT, NORM_X
	LREAL	x		
	TIME	-	No explicit conversion	-
	DTL	-		
	TOD	-		
	DATE	x	The bit pattern of the source value is transferred unchanged to the destination data type.	T_CONV, CONVERT2)
	STRING	x	The value is converted into a character string. If the length of the character string is violated, the enable output ENO is set to "0".	S_CONV, VAL_STRG, CONVERT2)
	CHAR1)	x	The bit pattern of the source value is transferred unchanged to the destination data type. If an overflow occurs, the enable output ENO is set to "0".	CONVERT

Source	Destination	Conversion	Description	Mnemonics of the instruction
x: Conversion possible				
- : Conversion not possible				
[1] Bit strings (BYTE, WORD, DWORD) and the CHAR data type are interpreted as unsigned integer with the same bit length. The BYTE data type is interpreted as USINT, WORD as UINT, and DWORD as UDINT. The CHAR data type is interpreted as USINT.				
[2] Is valid for SCL only				

See also

UINT (16-bit integers) (Page 752)

Implicit conversion of UINT (Page 790)

Overview of data type conversion (Page 780)

Explicit conversion of DINT

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the DINT data type. If a negative value is converted into an unsigned destination data type or if an overflow occurs, the enable output ENO is set to "0".

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DINT	BOOL[2]	x	The lowest bit is transferred into the destination data type.	CONVERT
	BYTE[1]	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	
	WORD[1]	x		
	DWORD[1]	x		
	SINT	x		
	USINT	x		
	INT	x		
	UINT	x		
	UDINT	x		
	REAL	x	The value is converted into the format of the destination data type (the value "-1", for example, is converted with the "CONVERT" instruction into the value "-1.0"). The result of the conversion depends on the mode of functioning of the instruction used.	CONVERT, NORM_X
	LREAL	x		
	TIME	x	The bit pattern of the source value is transferred unchanged to the destination data type.	T_CONV, CONVERT[2]
	DTL	-	No explicit conversion	-
	TOD	-		

Source	Destination	Conversio n	Explanation	Mnemonics of the instruction
	DATE	-		
	STRING	x	The value is converted into a character string. The character string is shown preceded by a sign. If the length of the character string is violated, the enable output ENO is set to "0".	S_CONV, VAL_STRG, CONVERT[2)
	CHAR[1)	x	The bit pattern of the source value is transferred unchanged to the destination data type.	CONVERT
	BCD	-	No explicit conversion	-
	BCD_DWORD	-		

x: Conversion possible

- : Conversion not possible

[1) Bit strings (BYTE, WORD, DWORD) and the CHAR data type are interpreted as unsigned integer with the same bit length. The BYTE data type is interpreted as USINT, WORD as UINT, and DWORD as UDINT. The CHAR data type is interpreted as USINT.

[2) Applies to SCL only

See also

DINT (32-bit integers) (Page 753)

Implicit conversion of DINT (Page 791)

Overview of data type conversion (Page 780)

Explicit conversion of UDINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the UDINT data type. If the permitted value range of the destination data type is violated, the enable out ENO is set to "0". In this case the result of the conversion is invalid.

Source	Destination	Conversion	Description	Mnemonics of the instruction
UDINT	BOOL[2)	x	The lowest bit is transferred into the destination data type.	CONVERT
	BYTE[1)	x	The bit pattern of the source value is transferred unchanged to the destination data type.	
	WORD[1)	x		
	DWORD[1)	x		
	SINT	x		
	USINT	x		
	UINT	x		
	INT	x		

Source	Destination	Conversion	Description	Mnemonics of the instruction
	DINT	x	The bit pattern of the source value is transferred unchanged to the destination data type. If the sign bit is overwritten during the conversion, the enable output ENO is set to "0".	
	REAL	x	The value is converted into the format of the destination data type (the value "1", for example, is converted with "Convert" instruction into the value "1.0"). The result of the conversion depends on the mode of functioning of the instruction used.	CONVERT, NORM_X
	LREAL	x		
	TIME	-	No explicit conversion	-
	DTL	-		
	TOD	x	The bit pattern of the source value is transferred unchanged to the destination data type.	T_CONV, CONVERT[2)]
	DATE	-	No explicit conversion	-
	STRING	x	The value is converted into a character string. If the length of the character string is violated, the enable output ENO is set to "0".	S_CONV, VAL_STRG, CONVERT[2)]
	CHAR[1)]	x	The bit pattern of the source value is transferred unchanged to the destination data type.	CONVERT

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD) and the CHAR data type are interpreted as unsigned integer with the same bit length. The BYTE data type is interpreted as USINT, WORD as UINT, and DWORD as UDINT. The CHAR data type is interpreted as USINT.

[2)] Is valid for SCL only

See also

UDINT (32-bit integers) (Page 754)

Implicit conversion of UDINT (Page 792)

Overview of data type conversion (Page 780)

Floating-point numbers

Explicit conversion of REAL

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of REAL data type:

Source	Destination	Conversion	Description	Mnemonics of the instruction
REAL	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	x [1]	The bit pattern of the source value is transferred unchanged to the destination data type.	CONVERT
	SINT	x	The value is converted into the destination data type. The result of the conversion depends on the instruction used. If the permitted value range of the destination data type is violated during the conversion or if the converted value is an invalid floating-point number, the enable output ENO is set to "0".	CONVERT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	USINT	x		
	INT	x		
	UINT	x		
	DINT	x		
	UDINT	x		
	LREAL	x	The value is converted into the destination data type.	
	TIME	-	No explicit conversion	-
	DTL	-		
	TOD	-		
	DATE	-		
	STRING	x	The value is converted into a character string. If the length of the character string is violated or if the value to be converted is an invalid floating-point number, the enable output ENO is set to "0".	S_CONV, VAL_STRG, CONVERT[1]
	CHAR	-	No explicit conversion	-
x: Conversion possible -: Conversion not possible [1] Is valid for SCL only				

See also

REAL (Page 755)

Implicit conversion of REAL (Page 793)

Overview of data type conversion (Page 780)

Explicit conversion of LREAL

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of LREAL data type:

Source	Destination	Conversion	Description	Mnemonics of the instruction
LREAL	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	-		
	SINT	x	The value is converted into the destination data type. The result of the conversion depends on the instruction used. If the permitted value range of the destination data type is violated during the conversion or if the value to be converted is an invalid floating-point number, the enable output ENO is set to "0".	CONVERT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	USINT	x		
	INT	x		
	UINT	x		
	DINT	x		
	UDINT	x		
	REAL	x	The value is converted into the destination data type. If the permitted value range of the destination data type is violated during the conversion or if the value to be converted is an invalid floating-point number, the enable output ENO is set to "0".	
	TIME	-	No explicit conversion	-
	DTL	-		
	TOD	-		
	DATE	-		
	STRING	x	The value is converted into a character string. If the length of the character string is violated or if the value to be converted is an invalid floating-point number, the enable output ENO is set to "0".	S_CONV, VAL_STRG, CONVERT[1]
	CHAR	-	No explicit conversion	-
x: Conversion possible -: Conversion not possible [1] Is valid for SCL only				

See also

LREAL (Page 756)

Overview of data type conversion (Page 780)

Timers

Explicit conversion of TIME

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of TIME data type:

Source	Destination	Conversion	Description	Mnemonics of the instruction
TIME	BYTE	-	No explicit conversion	-
	WORD	-		
	DWORD	-		
	SINT	-		
	USINT	-		
	INT	-		
	UINT	-		
	DINT	x	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in milliseconds.	T_CONV, CONVERT[1]
	UDINT	-	No explicit conversion	-
	REAL	-		
	LREAL	-		
	DTL	-		
	TOD	-		
	DATE	-		
	STRING	-		
	CHAR	-		
x: Conversion possible -: Conversion not possible [1] Is valid for SCL only				

See also

TIME (IEC time) (Page 757)

Implicit conversion of TIME (Page 794)

Overview of data type conversion (Page 780)

Clock and calendar

Explicit conversion of DATE

Options for explicit conversion

The DATE data type cannot be explicitly converted.

See also

DATE (Page 759)

Implicit conversion of DATE (Page 796)

Overview of data type conversion (Page 780)

Explicit conversion of TOD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of TOD data type:

Source	Destination	Conversion	Description	Mnemonics of the instruction
TOD	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	-		
	SINT	-		
	USINT	-		
	INT	-		
	UINT	-		
	DINT	-		
	UDINT	x	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00 hrs).	T_CONV, CONVERT[1)]
	REAL	-	No explicit conversion	-
	LREAL	-		
	TIME	-		
	DTL	-		
	DATE	-		
	STRING	-		
	CHAR	-		

Source	Destination	Conversion	Description	Mnemonics of the instruction
x: Conversion possible -: Conversion not possible [1] Is valid for SCL only				

See also

TIME_OF_DAY (TOD) (Page 760)

Implicit conversion of TOD (Page 795)

Overview of data type conversion (Page 780)

Explicit conversion of DTL

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of DTL data type:

Source	Destination	Conversion	Description	Mnemonics of the instruction
DTL	BYTE	-	No explicit conversion	-
	WORD	-		
	DWORD	-		
	SINT	-		
	USINT	-		
	INT	-		
	UINT	-		
	DINT	-		
	UDINT	-		
	REAL	-		
	LREAL	-		
	TIME	-		
	TOD	x	During the conversion, the time information is extracted from the DTL format and written to the destination data type.	T_CONV, CONVERT[1]
	DATE	x	During the conversion, the date information is extracted from the DTL format and written to the destination data type. If an overflow occurs, the enable output ENO is set to "0".	
	STRING	-	No explicit conversion	-
	CHAR	-		

Source	Destination	Conversion	Description	Mnemonics of the instruction
x: Conversion possible -: Conversion not possible 1) Is valid for SCL only				

See also

DTL (Page 762)

Overview of data type conversion (Page 780)

Character strings

Explicit conversion of CHAR

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of CHAR data type:

Source	Destination	Conversion	Description	Mnemonics of the instruction
CHAR	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	-		
	SINT	x	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	CONVERT
	USINT	x		
	INT	x		
	UINT	x		
	DINT	x		
	UDINT	x		
	REAL	-	No explicit conversion	-
	LREAL	-		
	TIME	-		
	DTL	-		
	TOD	-		
	DATE	-		

Source	Destination	Conversion	Description	Mnemonics of the instruction
	STRING	x	The value is converted in the first character of the character string. If the length of the character string is not defined, the length "1" is set after the conversion. If the length of the character string is defined, this remains unchanged after the conversion.	S_CONV, CONVERT[1)]
x: Conversion possible - : Conversion not possible [1)] Is valid for SCL only				

See also

CHAR (character) (Page 763)

Implicit conversion of CHAR (Page 797)

Overview of data type conversion (Page 780)

Explicit conversion of STRING

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of STRING data type:

Source	Destination	Conversion	Description	Mnemonics of the instruction
STRING	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	-		
	SINT	x	The conversion starts with the first character of the string and ends with the end of the string or the first character that is invalid. The following characters are permitted for conversion: • Digit • Sign	S_CONV, STRG_VAL, CONVERT[1)]
	USINT	x		
	INT	x		
	UINT	x		
	DINT	x		
	UDINT	x		
	REAL	x		

Source	Destination	Conversion	Description	Mnemonics of the instruction
	LREAL	x	• Dot The first character of the string may be a sign (+, -) or a number. Leading spaces will be ignored. The dot is used as separation for the conversion of floating-point numbers. The exponential notation "e" or "E" is not permitted. The comma as thousand separator is permitted to the left of the decimal point but will be ignored. If the structure of the string is invalid for the conversion or if an overflow occurs, then the enable output ENO will be set to "0".	
	TIME	-	No explicit conversion	-
	DTL	-		
	TOD	-		
	DATE	-		
	CHAR	x	The first character of the string is transferred to the destination data type. If the string is empty, then the value "0" will be written in the destination data type.	S_CONV, CONVERT[1)]
x: Conversion possible - : Conversion not possible [1)] Is valid for SCL only				

See also

STRING (Page 763)

Overview of data type conversion (Page 780)

Additional conversion functions

Additional explicit conversion functions

Additional options for explicit conversion in SCL

The following table shows the additional options and instructions for explicit conversion into SCL:

Source	Destination	Description	Mnemonics of the instruction
WORD	BLOCK_DB	The bit pattern of WORD is interpreted as a data block number.	WORD_TO_BLOCK_DB
BLOCK_DB	WORD	The data block number is interpreted as a bit pattern of WORD.	BLOCK_DB_TO_WORD

9.1.1.6 Program flow control

EN/ENO mechanism

Basics of the EN/ENO mechanism

Introduction

Runtime errors that require a program abort can occur during the processing of instructions. You can use the EN/ENO mechanism to avoid such program aborts. This mechanism can be used at two levels:

- EN/ENO mechanism for individual instructions
- EN/ENO mechanism for complete blocks

EN/ENO mechanism for basic instructions

In LAD and FBD, certain instructions have an enable input EN and an enable output ENO.

You can use the enable input EN to make the execution of the instruction dependent on conditions. The instructions are only executed if the signal state is "1" at the enable input EN.

You can use the enable output ENO to query runtime errors in instructions and react to these.

The enable output ENO returns the signal state "1" if one of the following conditions applies:

- No error occurred during processing.

The enable output ENO returns signal state "0" if one of the following conditions applies:

- The EN input has signal state "0".
- An error occurred during processing.

The EN/ENO mechanism is used for the following basic instructions:

- Math functions
- Move operations
- Conversion operations
- Word logic operations
- Shift + rotate

With SCL, the use of the EN/ENO mechanism for instructions is optional. You can activate it with the block property "Set ENO automatically".

In STL, the EN/ENO mechanism is not required for individual instructions. This function is mapped there by language-specific instruction sequences.

EN/ENO mechanism for block calls

When called, all blocks are provided with a enable input EN and an enable output ENO. You can use the enable input EN to call the block depending on conditions. The block is only executed if the signal state is "1" at the enable input EN.

The enable output ENO has signal "1" as soon as the execution of the block starts. If one of the instructions within the block generates a runtime error, ENO is set to "0". You can also use ENO as group error message, for example.

Note

When you call functions in SCL, you cannot use the release mechanism via EN. Use an IF statement instead to call functions conditionally.

See also

Example of the use of the EN/ENO mechanism in LAD (Page 821)

Example of the simulation of the EN/ENO mechanism in STL (Page 823)

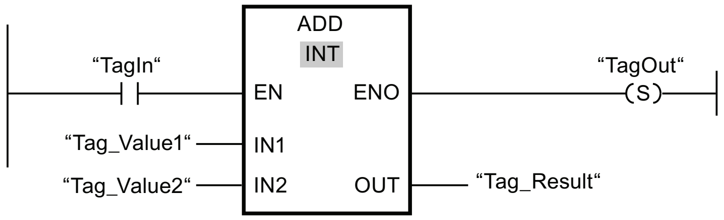
Example of the use of the EN/ENO mechanism in FBD (Page 821)

Example of the use of the EN/ENO mechanism in SCL (Page 822)

Example of the use of the EN/ENO mechanism in LAD

Description

The following figure shows an ADD instruction with EN and ENO protective circuit:



After the normally open contact, the RLO contains the previous result of logic operation:

- If "TagIn" signal is "0", the addition is not executed. EN and ENO both lead to the signal state "0".
- If "TagIn" signal is "1", EN is also "1" and the addition is executed. If no errors occur during the processing of the instruction, the output ENO also has the signal state "1" and the output ""TagOut"" is set.

See also

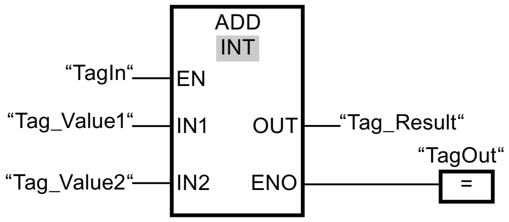
Basics of the EN/ENO mechanism (Page 819)

ADD: Add (Page 1289)

Example of the use of the EN/ENO mechanism in FBD

Description

The following figure shows an ADD instruction with EN and ENO protective circuit:



- If "TagIn" signal is "1", EN is also "1" and the addition is executed. If no errors occur during the processing of the instruction, the output ENO also has the signal state "1" and the output ""TagOut"" is set.
- If "TagIn" signal is "0", the addition is not executed. EN and ENO both lead to the signal state "0".

See also

Basics of the EN/ENO mechanism (Page 819)

Example of the use of the EN/ENO mechanism in SCL

Example of the EN/ENO mechanism for basic instructions

To use the EN/ENO mechanism for instructions in SCL, you have to activate the block property "Set ENO automatically". The following example shows the use of the enable output ENO for the "a/b" instruction.

SCL

```
"MyoutputREAL" :=a/b;
IF ENO
 THEN "MyOutputBool":=1;
 ELSE "MyOutputBool":=0;
END_IF;
```

If the "a/b" instruction is executed error-free, MyOutputBool has signal "1".

Example of the use of the EN/ENO mechanism in block calls

The following example shows the use of the enable output ENO for a block call.

SCL

```
"MyDB"."MyFB"(EN:="MyTag1">"MyTag2",
 in1:="MyInputBool1",
 in2:="MyInputBool1",
 EN0=>"MyOutputBool");
```

If MyTag1 is not greater than MyTag2 the block call is not processed. EN and ENO both lead to the signal state "0".

If MyTag1 is greater than MyTag2, EN has signal "1" and the block call is executed.

If all instructions within MyFB are executed error-free, MyOutputBool has signal "1".

See also

Basics of the EN/ENO mechanism (Page 819)

Example of the simulation of the EN/ENO mechanism in STL

Description

The following example shows an program section for adding values with EN and ENO connected:

STL	Description
A"Tag_Input_1"	// Query whether the signal state of the operand is "1" and AND with current RLO
JNBMyLABEL	// Evaluation of the EN input
	// If RLO="0" jump to jump label "MyLABEL" and save the current RLO in the BR
	// Execute next instruction if RLO="1"
L"Tag_Input_2"	// Load first value of addition
L"Tag_Input_3"	// Load second value of addition
+I	// Add values
T "Tag_Result"	// Transfer sum to the operand "Tag_Result"
AN OV	// Query if errors occurred
SAVE	// Transfer signal state of the RLO to the BR bit
CLR	// Reset RLO to "0"
MyLABEL: U BR	// Jump label "MyLABEL"
	// Query BR and AND with RLO
= "Tag_Output"	// Assign signal state of the RLO to the operand "Tag_Output"

The query of the operand "U" Tag_Input_1"" provides the result of the preceding logic operation (RLO). The instruction "Jump at RLO = 0 and save RLO (SPBNB)" saves the RLO to the BR. The instruction "Jump if RLO = 0 and save RL0" also evaluates the RLO and executes one of the following actions depending on the evaluation:

- If the RLO is "0", the processing of the program is continued at the jump label "MyLABEL" with the query of the BR. The addition is not executed. Assign the current RLO to the operand "Tag_Output".
- If the RLO is "1", the addition is executed. A query of the overflow bit (OV) shows if an error occurred during the addition. The query result is saved in the BR. The operation "CLR" resets the RLO to "0". The BR is then queried for "1" and AND'd with the current RLO. The result is assigned to the operand "Tag_Output". The signal state of the BR and of the operand "Tag_Output" shows if the addition was carried out with any error

See also

Basics of the EN/ENO mechanism (Page 819)

9.1.2 Declaring PLC tags

9.1.2.1 Overview of PLC tag tables

Introduction

PLC tag tables contain the definitions of the PLC tags and symbolic constants that are valid throughout the CPU. A PLC tag table is created automatically for each CPU used in the project. You can create additional tag tables and use these to sort and group tags and constants.

In the project tree there is a "PLC tags" folder for each CPU of the project. The following tables are included:

- "All tags" table
- Standard tag table
- Optional: Other user-defined tag tables

All tags

The "All tags" table gives an overview of all PLC tags, user constants and system constants of the CPU. This table cannot be deleted or moved.

Standard tag table

There is one standard tag table for each CPU of the project. It cannot be deleted, renamed or moved. The default tag table contains PLC tags, user constants and system constants. You can declare all PLC tags in the default tag table, or create additional user-defined tag tables as you want.

User-defined tag tables

You can create multiple user-defined tag tables for each CPU to group tags according to your requirements. You can rename, gather into groups, or delete user-defined tag tables. User-defined tag tables can contain PLC tags and user constants.

See also

Structure of the PLC tag tables (Page 825)

Using tags within the program (Page 717)

Constants (Page 718)

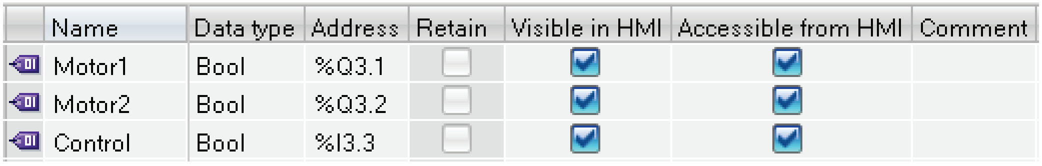
9.1.2.2 Structure of the PLC tag tables

Introduction

Each PLC tag table contains a tab for tags and a tab for user constants. The default tag table and the "All tags" table also have a "System constants" tab.

Structure of the "PLC tags" tab

In the "Tags" tab you declare the global PLC tags that you require in the program. The following figure shows the tab structure. The number of columns shown may vary.



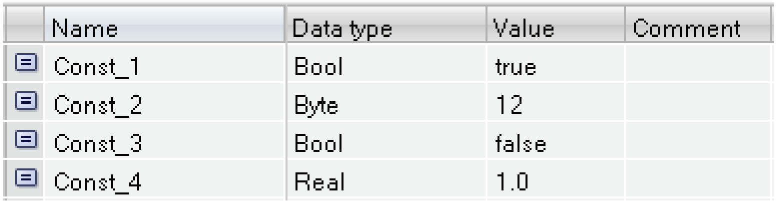
	Name	Data type	Address	Retain	Visible in HMI	Accessible from HMI	Comment
	Motor1	Bool	%Q3.1	□	☑	☑	
	Motor2	Bool	%Q3.2	□	☑	☑	
	Control	Bool	%I3.3	□	☑	☑	

The following table shows the meaning of the individual columns. The number of columns shown may vary. You can show or hide the columns as required.

Column	Description
	Symbol you can click on to drag-and-drop a tag to a program for use as an operand.
Name	Unique name for the constants throughout the CPU.
Data type	Data type of the tags.
Address	Tag address.
Retain	Marks the tag as retentive. The values of retentive tags are retained even after the power supply is switched off.
Accessible from HMI	Shows whether HMI can access this tag during runtime.
Visible in HMI	Shows whether the tag is visible by default in the operand selection of HMI.
Monitor value	Current data value in the CPU. This column only appears if an online connection is available and you select the "Monitor" button.
Tag table	Shows which tag table includes the tag declaration. This column is only available in the "All tags" table.
Comment	Comments to document the tags.

Structure of the "User constants" and "System constants" tabs

In the "User constants" you define symbolic constants that are valid throughout the CPU. The constants required by the system are shown in the "Systems constants" tab. The following figure shows the structure of both tabs. The number of columns shown may vary.



	Name	Data type	Value	Comment
▤	Const_1	Bool	true	
▤	Const_2	Byte	12	
▤	Const_3	Bool	false	
▤	Const_4	Real	1.0	

The following table shows the meaning of the individual columns. You can show or hide the columns as required.

Column	Description
▤	Symbol you can click to move a tag into a network via a drag-and-drop operation for use as an operand.
Name	Unique name for the constants throughout the CPU.
Data type	Data type of the constants
Value	Value of the constants
Tag table	Shows which tag table includes the constant declaration. This column is only available in the "All tags" table.
Comment	Comments to document the tags.

See also

Using tags within the program (Page 717)

Constants (Page 718)

Overview of PLC tag tables (Page 824)

Show and hide table columns (Page 842)

9.1.2.3 Creating and managing PLC tag tables

Creating a PLC tag table

You can created multiple user-defined PLC tag tables in a CPU. Each tag table must have have a unique name throughout the CPU.

Requirement

The project view is open.

Procedure

To created a new PLC tag table, follow these steps:

1. Open the "PLC tags" folder under the CPU in the project tree.
2. Double-click the "Add new tag table" entry.

   A new PLC tag table with the standard name "TagTable_x" is opened.
3. Select the PLC tag table in the project tree.
4. Select the "Rename" command in the shortcut menu.
5. Type in a name that is unique throughout the CPU.

Result

A new PLC tag table is created. You can now declare tag and constants in this table.

See also

Overview of PLC tag tables (Page 824)

Structure of the PLC tag tables (Page 825)

Importing and exporting (Page 1200)

Grouping PLC tag tables

You can gather the user-defined tag tables of the CPU into groups. You cannot, however, move the standard tag table and the "All tags" table into a group.

Requirement

Multiple user-defined tag tables are contained in the "PLC tags" folder of the CPU.

Procedure

To gather multiple PLC tag tables into a group, follow these steps:

1. Select the "PLC tags" folder under the CPU in the project tree.
2. Select the "Insert > Group" menu command.

   A new group with the standard name "Group_x" is inserted.
3. Select the newly inserted group in the project tree.
4. Select the "Rename" command in the shortcut menu.
5. Assign the new group a unique name throughout the CPU.
6. Drag to the new group the tables you want to group together.

Result

The tag tables are gathered in the new group.

See also

Overview of PLC tag tables (Page 824)

Structure of the PLC tag tables (Page 825)

Opening the PLC tag table

Procedure

To open the PLC tag table in a CPU, proceed as follows:

1. Open the "PLC tags" folder under the CPU in the project tree.
2. Double-click the PLC tag table in the folder.
3. Select the desired tab in the upper corner.

Result

The PLC tag table associated with the CPU opens. You can declare the required tags and constants.

See also

Overview of PLC tag tables (Page 824)

Structure of the PLC tag tables (Page 825)

9.1.2.4 Declaring PLC tags

Rules for PLC tags

Valid names of PLC tags

Permissible characters

The following rules apply to the use of names for PLC tags:

- Letters, numbers, special characters are permitted.
- Quotation marks are not permitted.

Unique tag names

The names of the PLC tags must be unique throughout the CPU, even if the tags are located in different tag tables of a CPU. A name that is already used for a block, another PLC tag or a constant within the CPU, cannot be used for a new PLC tag. The uniqueness check does not differentiate between use of small and capital letters.

If you enter an already assigned name another time, a sequential number is automatically appended to the second name entered. For example, if you enter the name "Motor" a second time, the second entry is changed to "Motor(1)".

Unique table names

The names of the PLC tag tables must also be unique throughout the CPU. A unique name is automatically suggested when user-defined PLC tag tables are being created.

See also

Using tags within the program (Page 717)

Permissible addresses and data types of PLC tags (Page 829)

Reserved key words (Page 719)

Permissible addresses and data types of PLC tags

The addresses of PLC tags are made up of the particulars of the operand area and the address within this area.

The addresses must be unique throughout the CPU. If you enter an address that is already assigned to another tag, the address will be highlighted at both places in yellow and an error message will be issued.

Operand areas

The following table shows the possible operand areas. The available data types depend on the CPU you use:

Operand area		Description	Data type	Format	Address area:	
International mnemonics	German mnemonics				S7-1200	S7-300/400
I	E	Input bit	BOOL	I x.y E x.y	0.0..1023.7	0.0..65535.7
IB	EB	Input byte	BYTE, CHAR, SINT, USINT	IB x EB y	0..1023	0..65535
IW	EW	Input word	WORD, INT, UINT, DATE, S5TIME	IW x EW y	0..1022	0..65534
ID	ED	Input double word	DWORD, DINT, UDINT, REAL, TIME, TOD	ID x ED y	0..1020	0..65532
Q	A	Output bit	BOOL	Q x.y A x.y	0.0..1023.7	0.0..65535.7
QB	AB	Output byte	BYTE, CHAR, SINT, USINT	QB x AB y	0..1023	0..65535
QW	AW	Output word	WORD, INT, UINT, DATE, S5TIME	QW x AW y	0..1022	0..65534
QD	AD	Output double word	DWORD, DINT, UDINT, REAL, TIME, TOD	QD x AD y	0..1020	0..65532
M	M	Memory bit	BOOL	M x.y	0.0..8191.7	0.0..65535.7
M	M	Memory bit	LREAL	M x.0	0.0..8190.0	-
MB	MB	Memory byte	BYTE, CHAR, SINT, USINT	MB x	0..8191	0..65535
MW	MW	Memory word	WORD, INT, UINT, DATE, S5TIME	MW x	0..8190	0..65534
MD	MD	Memory double word	DWORD, DINT, UDINT, REAL, TIME, TOD	MD x	0..8188	0..65532
T	T	Time function (for S7-300/400 only)	Timer	T n	-	0..65535
C	C	Counter function (for S7-300/400 only)	Counter	Z n C n	-	0..65535

Addresses

The following table shows the possible addresses of tags:

Data type	Address	Example
BOOL	Tags with BOOL data type are addressed with a byte number and a bit number. The numbering of the bytes begins for each operand area at 0. The numbering of the bits goes from 0 to 7.	A 1.0
BYTE, CHAR, SINT, USINT	Tags with BYTE, CHAR, SINT, and USINT data type are addressed with a byte number.	MB 1
WORD, INT, UINT, DATE	Tags with WORD, INT, UINT, DATE data type consist of two bytes. They are addressed with the number of the lowest byte.	IW 1
DWORD, DINT, UDINT, REAL, TIME	Tags with DWORD, DINT, UDINT, REAL, TIME data type are made up of four bytes. They are addressed with the number of the lowest byte.	AD 1

Mnemonics used

The addresses that you enter in the PLC tag table are automatically adapted to the set mnemonics.

See also

Setting the mnemonics (Page 895)

Using tags within the program (Page 717)

Valid names of PLC tags (Page 829)

Overview of the valid data types (Page 741)

Entering a PLC tag declaration

Declaring tags in the PLC tag table

Requirements

The "Tags" tab of the PLC tag table is open.

Procedure

To define PLC tags, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the required data type in the "Data type" column. You will be supported by autocompletion during input.

   An address corresponding to the data type is automatically appended.
3. Optional: Click on the arrow key in the "Address" column and enter an operand identifier, an operand type, an address and a bit number in the dialog which then opens.
4. Optional: Enter a comment in the "Comments" column.
5. Repeat steps 1 to 4 for all the tags you require.

See also: Permissible addresses and data types of PLC tags (Page 829)

Syntax check

A syntax check is performed automatically after each entry, and any errors found are displayed in red. You do not have to correct these errors immediately - you can continue editing and make any corrections later. As long as the tag declaration contains syntax errors and the tag is used in the program, you will not be able to compile the program.

See also

Importing and exporting (Page 1200)

Valid names of PLC tags (Page 829)

Declaring PLC tags in the program editor (Page 832)

Structure of the PLC tag tables (Page 825)

Declaring PLC tags in the program editor

Requirement

- The program editor is open.

Procedure

To declare operands as global PLC tags, follow these steps:

1. Insert an instruction in your program.

   The "<???>", "<??.?>" or "..." strings represent operand placeholders.

2. Replace an operand placeholder with the name of the PLC tag to be created.

3. Select the tag name.

   If you want to declare multiple PLC tags, select the names of all the tags to be declared.

4. Select the "Define tag" command in the shortcut menu.

   The "Define tag" dialog box opens. This dialog displays a declaration table in which the name of the tag is already entered.

5. Click the arrow key in the "Section" column and select one of the following entries:

   – Global Memory

   – Global Input

   – Global Output

6. In the other columns, enter the address, data type, and comments.

   See also: Permissible addresses and data types of PLC tags (Page 829)

7. If the CPU contains multiple PLC tag tables, you can use an entry in the "PLC tag table" column to indicate in which table the tag is to be inserted. If you make no entry in the column, the new tag will be inserted in the default tag table.

8. Click the "Define" button to complete your entry.

Result

The tag declaration is written to the PLC tag table and is valid for all blocks in the CPU.

See also

Valid names of PLC tags (Page 829)

Declaring tags in the PLC tag table (Page 831)

Setting the retentivity of PLC tags

Retentive behavior of PLC tags

Retentive memory areas for PLC tags

To prevent data loss in the event of power failure, you can define a part of the CPU's system memory as retentive. Values of PLC tags that tile in the retentive memory area are preserved even when the power supply is switched off.

You specify the exact width of the retentive memory area in the PLC tag table.

See also

Setting the retentive behavior of PLC tags (Page 834)

Setting the retentive behavior of PLC tags

Introduction

In the PLC tag table you can specify the width of the retentive memory area for PLC tags. All tags with addresses in this memory area are then designated as retentive. You can recognize the retentivity setting of a tag by the check mark set in the "Retain" column of the PLC tag table.

Requirement

The "PLC tags" tab of the PLC tag table is open.

Procedure

To define the width of the retentive memory area for PLC tags, follow these steps:

1. On the toolbar, click the "Retain" button.

   The "Retain memory" dialog will open.
2. Specify the width of the retentive memory area by entering the number of memory bytes in the input field.
3. Click the "OK" button.

Result

The width if the retentive memory area is defined. In the "Retain" column of the tag table a check mark is automatically set for all tags that are located within the retentive memory area.

See also

Retentive behavior of PLC tags (Page 834)

9.1.2.5 Declaring symbolic constants

Rules for symbolic constants

Permissible characters

Names of symbol constants may consist of the following characters:

- Letters, numbers, special characters are permitted.
- Quotation marks are not permitted.

Unique constant names

The names of the symbolic constants must be unique throughout the CPU, even if the constants are located in different tag tables of a CPU. A name that is already used for a block, a PLC tag or another constant within the CPU, cannot be used for new constant. The uniqueness check does not differentiate between use of small and capital letters.

If you enter an already assigned name another time, a sequential number is automatically appended to the second name entered. For example, if you enter the name "Motor" a second time, the second entry is changed to "Motor(1)".

Permissible data types

For constants, all data types supported by the CPU are permitted, with the exception of structured data types.

Permitted values

You can select any value from the value range of the specified data type as constant value. For information on the value ranges, refer to the "Data types" chapter.

See also: Data types (Page 741)

See also

Constants (Page 718)

Declaring constants (Page 835)

Declaring constants

Introduction

You declare constants in the "User constants" tab of a PLC tag table. During declaration you have to to enter a symbolic name, a data type and a fixed value for each constant. The entry format and the value range of the constant value depend on the data type of the constant.

See also: Data types (Page 741)

Procedure

To declare constants, follow these steps:

1. Open a PLC tag table.
2. Open the "User constants" tab.

   The constants table opens.
3. Enter a constant name in the "Name" column.
4. Enter the required data type in the "Data type" column. You will be supported by autocompletion during input.
5. Enter a constant value in the "Value" column; this constant value must be valid for the selected data type.
6. If you want, enter comments on the constants in the "Comments" column. The entry of a comment is optional.
7. If you want to declare additional constants, place the cursor in the next row and repeat steps 3 to 6.

Syntax check

A syntax check is performed automatically after each entry, and any errors found are displayed in red. You do not have to correct these errors immediately - you can continue editing and make any corrections later. As long as the tag declaration contains syntax errors and the constant is used in the program, you will not be able to compile the program.

See also

Opening the PLC tag table (Page 828)

Inserting a table row in the PLC tag table (Page 840)

Structure of the PLC tag tables (Page 825)

Rules for symbolic constants (Page 835)

9.1.2.6 Editing properties

Editing the properties of PLC tags

Properties of PLC tags

Overview

The following table gives an overview of the properties of PLC tags:

Group	Property	Description
General	Name	A unique name within the CPU.
	Data type	Data type of the tags.
	Address	Tag address.
	Comment	Comment on the tag.
Time stamp	Date created	Time when the tag was created (cannot be changed).
	Last modified	Time when the tag was last changed (cannot be changed).
Usage	Visible in HMI	Shows whether the tag is visible by default in the HMI selection list.
	Accessible from HMI	Shows whether HMI can access this tag during runtime.

See also

Editing the properties of PLC tags (Page 837)

Editing the properties of PLC tags

Editing properties in a PLC tag table

To edit the properties of one or more tags, follow these steps:

1. In the project tree, double-click the PLC tag table that contains the tags.

   The PLC tag table opens.

2. Change the entries in the columns.

Editing addresses in the program editor

To edit the address of a tag in the program editor, follow these steps:

1. Select the tag name.
2. Select the "Rewire tag" command in the shortcut menu.

   The "Rewire tag" dialog will open. The dialog shows a declaration table.
3. Enter the new address in the "Address" column.
4. Click the "Change" button to confirm the input.

Editing names in the program editor

To edit the name of a tag in the program editor, follow these steps:

1. Select the tag name.
2. Select the "Rename tag" command in the shortcut menu.

   The "Rename tag" dialog opens. The dialog shows a declaration table.
3. Enter the new name in the "Name" column.
4. Click the "Change" button to confirm the input.

Effect in the program

In the case of a change of the tag's name, data type or address, each location of use of the tag is automatically updated in the program.

See also

Properties of PLC tags (Page 837)

Editing the properties of symbolic constants

Properties of constants

Overview

The following table gives an overview of the properties of constants:

Group	Property	Description
General	Name	A unique name within the table
	Data type	Data type of the constants
	Value	Value that you defined for the constants. This value must be compatible with the declared data type. See also: Data types (Page 741)
	Comment	Comment on the constants
History	Date created	Time when the constant was created (cannot be changed)
	Last modified	Time when the constant was last changed (cannot be changed)

Editing properties of constants

Editing properties in a PLC tag table

To edit the properties of one or more constants, follow these steps:

1. In the project tree, double-click the PLC tag table that contains the constants.

   The PLC tag table opens.
2. Open the "User constants" tab.
3. Change the entries in the "Name", "Data type", "Value", or "Comments" column.

Effect in the program

In the case of a change of a constant's name, data type or value, each location of use of the constant is automatically updated in the program.

See also

Data types (Page 741)

9.1.2.7 Monitoring of PLC tags

Monitoring of PLC tags

You can monitor the current data values of the tags on the CPU directly in the PLC tag table.

Requirements

An online connection to the CPU is available.

Procedure

To monitor the data values, proceed as follows:

1. Open a PLC tag table.
2. Start monitoring by clicking the "Monitor all" button.

   The additional "Monitor value" column is displayed in the table. This shows the current data values.
3. End the monitoring by clicking the "Monitor all" button again.

Note

You also have the option of copying PLC tags to a monitor or force table so that you can monitor, control or force them in the table.

See also

Structure of the PLC tag tables (Page 825)

Introduction to testing with the watch table (Page 1133)

Introduction for testing with the force table (Page 1156)

Copying entries in the PLC tag table (Page 841)

9.1.2.8 Editing PLC tag tables

Inserting a table row in the PLC tag table

Procedure

Proceed as follows to insert a row above the selected row:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button on the toolbar of the table.

Result

A new row is inserted above the selected row.

Copying entries in the PLC tag table

You can copy PLC tags within a table or to other tables.

Procedure

To copy a tag, follow these steps:

1. Select the tags you want to copy.

   You can also select several tags by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last tag.
2. Select "Copy" in the shortcut menu.
3. Position the insertion pointer at the location where you want to insert the tags.
4. Select "Paste" in the shortcut menu.

Or

1. Select the tag.
2. Hold down the left mouse button.
3. At the same time, press <Ctrl>.
4. Drag the tag to the destination.

Result

- The tag is copied to the destination.
- If there is a name conflict, a number is automatically appended to the tag name. For example, "Tag" becomes "Tag(1)".
- All other properties of the tag remain unchanged.

Deleting entries in the PLC tag table

Procedure

To delete a tag, follow these steps:

1. Select the row with the tag to be deleted. You can also select several rows by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last row.
2. Select the "Delete" command in the shortcut menu.

Sorting rows in the PLC tag tables

You can sort the rows in the tables alphanumerically by name, data type, or address.

Procedure

To sort the table rows, follow these steps:

1. Select the column by which you want to sort.
2. Click the column header.

   The column will be sorted in order of increasing values.

   An up arrow shows the sort sequence.
3. In order to change the sort sequence, click the arrow.

   The column will be sorted in order of decreasing values. A down arrow shows the sort sequence.
4. To restore the original sequence, click a third time on the column header.

Automatically filling in cells in the PLC tag table

You can load the contents of one or several table cells into the cells below, automatically filling in the successive cells.

If you automatically fill in cells in the "Name" column, a consecutive number will be appended to each name. For example, "Motor" will become "Motor_1".

If you fill the cells in the column "Address" automatically, the addresses will be increased depending on the indicated data type.

Procedure

To automatically fill in successive cells, follow these steps:

1. Select the cells to be loaded.
2. Click the "Fill" symbol in the bottom right corner of the cell.

   The mouse pointer is transformed into a crosshair.
3. Keep the mouse button pressed and drag the mouse pointer downwards over the cells that you want to fill in automatically.
4. Release the mouse button.

   The cells are filled in automatically. If entries are already present in the cells that are to be automatically updated, a dialog appears in which you can indicate whether you want to overwrite the existing entries or whether you want to insert new rows for the new tags.

Show and hide table columns

You can show or hide the columns in a table as needed.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. In the shortcut menu, select the "Show/hide columns" command.

   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.

Editing PLC tags with external editors

To edit individual PLC tags in external editors outside the TIA portal, you can export or import these tags using copy and paste. However, you cannot copy structured tags to an editor.

Requirement

A PLC tag table and an external editor are opened.

Procedure

To export and import individual PLC tags, follow these steps:

1. Select one or more PLC tags.
2. Select "Copy" in the shortcut menu.
3. Switch to the external editor and paste the copied tags.
4. Edit the tags as required.
5. Copy the tags in the external editor.
6. Switch back to the PLC tag table.
7. Select "Paste" in the shortcut menu.

Note

You also have the option of export or importing PLC tags as mass data.

See also: Importing and exporting (Page 1200)

9.1.3 Creating and managing blocks

9.1.3.1 Creating blocks

Block folder

Function

You can find a "Program blocks" folder in the project tree, in which you can create and manage the following blocks:

- Organization blocks (OB)
- Function blocks (FB)
- Functions (FCs)
- Data blocks (DB)

A "System blocks" subfolder containing another subfolder, "Program resources", is also created in the "Program blocks" folder the first time you drag an instruction to your program which is an internal system function block. The instance data block of the internal system function block is also pasted to the "Program resources" folder. You can move or copy such instance data blocks from the "Program resources" folder to any other folder and rename or delete them. You can also move your blocks into the "Program resources" folder. Blocks in the "Program resources" folder that are not required to run the user program are removed during the next compilation. If the "Program resources" folder contains no more blocks then it is also deleted with the "System blocks" folder.

A program cycle OB is automatically generated for each device and inserted in the "Program blocks" folder.

See also

Creating functions and function blocks (Page 845)

Creating data blocks (Page 846)

Creating organization blocks (Page 844)

Using blocks from libraries (Page 848)

Creating organization blocks

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To create an organization block, follow these steps:

1. Double-click the "Add new block" command.

   The "Add new block" dialog box opens.
2. Click the "Organization block (OB)" button.
3. Select the type of new organization block.
4. Enter a name for the new organization block.
5. Enter the properties of the new organization block.
6. To enter additional properties for the new organization block, click "Additional information".

   An area with further input fields is displayed.
7. Enter all the properties you require.
8. Activate the "Add new and open" check box if the organization block does not open as soon as it is created.
9. Confirm your entries with "OK".

Result

The new organization block is created. You can find the organization block in the project tree in the "Program blocks" folder. You can assign additional parameters to some organization blocks in the inspector window or device view after they have been created. The organization block description will state whether the newly created organization block has additional parameters.

See also

Organization blocks (OB) (Page 696)

Block folder (Page 844)

Creating functions and function blocks (Page 845)

Creating data blocks (Page 846)

Using blocks from libraries (Page 848)

Entering a block title (Page 851)

Entering a block comment (Page 852)

Creating functions and function blocks

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To create a function (FC) or a function block (FB), follow these steps:

1. Double-click the "Add new block" command.

   The "Add new block" dialog box opens.
2. Click the "Function block (FB)" or "Function (FC)" button.
3. Enter a name for the new block.
4. Enter the properties of the new block.
5. To enter additional properties for the new block, click "Additional information".

   An area with further input fields is displayed.
6. Enter all the properties you require.
7. Activate the "Add new and open" check box if the block does not open as soon as it is created.
8. Confirm your entries with "OK".

Result

The new block is created. You can find the block in the project tree in the "Program blocks" folder.

See also

Function blocks (FB) (Page 697)

Functions (FCs) (Page 697)

Basics of block access (Page 700)

Block folder (Page 844)

Creating organization blocks (Page 844)

Creating data blocks (Page 846)

Using blocks from libraries (Page 848)

Entering a block title (Page 851)

Entering a block comment (Page 852)

Creating data blocks

Requirements

The "Program blocks" folder in the project tree is open.

Procedure

To create a data block, follow these steps:

1. Double-click the "Add new block" command.

   The "Add new block" dialog box opens.

2. Click the "Data block (DB)" button.

3. Select the type of the data block. You have the following options available to you:

   – To create a global data block, select the list entry "Global DB".

   – To create an instance data block, select the function block to which you want to assign the instance data block from the list. The list contains only the function blocks that were previously created for the CPU.

   – To create a data block based on a PLC data type, select the PLC data type from the list. The list contains only the PLC data types that were previously created for the CPU.

   – To create a data block based on a system data type, select the system data type from the list. The list contains only those system data types that have already been inserted to program blocks in the CPU.

4. Enter a name for the data block.

5. Enter the properties of the new data block.

   – Select whether you want to assign the block number manually or automatically. Enter a number if you decide to use manual assignment.

   – Select the type of the block access (for S7-1200 only).

6. To enter additional properties of the new data block, click "Additional information".

   An area with further input fields is displayed.

7. Enter all the properties you require.

8. Activate the "Add new and open" check box if the block does not open as soon as it is created.

9. Confirm your entry with "OK".

Result

The new data block is created. You can find the data block in the project tree in the "Program blocks" folder.

See also

Global data blocks (DB) (Page 698)

Instance data blocks (Page 699)

Block folder (Page 844)

Creating organization blocks (Page 844)

Creating functions and function blocks (Page 845)

Using blocks from libraries (Page 848)

Basics of block access (Page 700)

System data types (Page 776)

Using blocks from libraries

You can save blocks in the project library or in a global library, so that you can use them more than once within a user program.

Requirement

- The "Libraries" task card is displayed.
- No write protection is set for global libraries.

Adding blocks to the project library or to a global library

To add new blocks to the project library or to a global library, follow these steps:

1. Maximize the project library or the global library.
2. Use drag-and-drop to move the block you wish to add to the library to the "Master copies" folder or any one of the "Master copies" subfolders in the project library or a global library. Don't release the left mouse button until a small plus sign appears underneath the mouse pointer.

Using blocks of the project library or a global library

To use a block from the project library or a global library in your project, follow these steps:

1. Maximize the project library or the global library so that you can see the block you wish to use.
2. Use a drag-and-drop operation to move the block to the CPU block folder. If the selected insertion points is not allowed, the mouse pointer will appear as a circle with a slash.

Copying and pasting blocks

Basics of copying and pasting blocks

Function

You can also create new blocks by copying existing blocks and pasting the copy. Observe the following principles when using this method:

- You can copy organization blocks (OBs), functions (FCs), function blocks (FBs), and global data blocks (DBs) without restriction.
- You can copy instance data blocks only for the same function block, since the assignment to the function block cannot be changed afterwards. However, the assignment is canceled if you copy the instance data block to a different CPU. If a function block with the same number exists there, the instance data block will be assigned to this function block. If you copy the instance data block together with the function block into the other CPU, the instance data block is assigned to the copy of the function block.

Copying data

With paste, all the block data is copied and forwarded to the copy. This data includes:

- Block interface tags
- All networks
- Comments in all existing compilations
- Messages defined in the block
- The entire program code of the copied block including the call instructions contained in the block.

  However, called blocks and their associated instance data blocks are not copied.

Avoiding name conflicts during pasting

When pasting copied blocks with identical names to already existing blocks, the following mechanisms are used to avoid name conflicts:

- Pasting the copied block into the same CPU:

  The copy of the block gets a name that is extended by a number. For example, if block "A" is copied, a possible name for the copy is "A_1". Consecutive numbering is not used, but rather the smallest free number. The copy of block "A" can also get the name "A_25", if no lower number is available.
- Pasting the copied block into another CPU:

  A dialog box opens in which you can select whether the block with the same name will be replaced or the copied block will be pasted with a duplicate designation (Name_Number).

NOTICE
Number conflicts may occur, if the pasted block has the same block number as an existing block. The block number is not automatically changed during pasting. This double number may have an effect on block calls. When you copy blocks you should therefore check the block number carefully and correct duplicate block numbers manually or using the block properties.

See also

Copying blocks (Page 850)

Pasting blocks (Page 850)

Copying blocks

Requirement

The "Program blocks" folder is opened in the project tree.

Procedure

To copy a block, follow these steps:

1. Right-click the block that you want to copy.
2. Select "Copy" in the shortcut menu.

Result

A copy of the block is now on the clipboard and can be pasted either into the same CPU or into another one.

See also

Basics of copying and pasting blocks (Page 849)

Pasting blocks (Page 850)

Pasting blocks

Requirement

You have copied a block.

Procedure

To paste a copied block and its data into a CPU, follow these steps:

1. In the project tree, open the folder structure for the CPU into which you want to paste the copied block.

   Note

   Please note that you can only paste the copied block into a CPU which supports the block programming language and type.

   ---

2. Right-click on the "Program blocks" folder.
3. Select "Paste" in the shortcut menu.
   – If you are pasting the block into the same CPU as the original block, "_<consecutive number>" will be appended to the name of the copy.
   – If you are pasting the block into a different CPU where a block of the same name already exists, the "Paste" dialog box opens. Select the required option and confirm with "OK".

See also

Basics of copying and pasting blocks (Page 849)

Copying blocks (Page 850)

Entering a block title

The block title is the title of the block. It is not the same thing as the block name, which was assigned when the block was created. The length of the block title is restricted to one row. You can enter the block title for open and closed blocks.

Requirement

A code block is available.

Enter block title for open block

To insert the block title in an open block, follow these steps:

1. Click on the title bar of the block in the program editor.
2. Enter the block title.

Enter block title for closed blocks

To insert the block title in a closed block, follow these steps:

1. Right-click the block in the project tree.
2. Select the "Properties" command in the shortcut menu.

   The dialog with the properties of the block opens.
3. Select the entry "Information" in the area navigation.
4. Enter the block title in the "Title" input field.
5. Confirm your entry with "OK".

See also

Creating organization blocks (Page 844)

Creating functions and function blocks (Page 845)

Entering a block comment (Page 852)

Entering a block comment

You can use the block comment to document the entire code block. For example, you can indicate the purpose of the block or draw attention to special characteristics. You can enter the block comment for open and closed blocks.

Requirement

A code block is available.

Enter block comment for open blocks

To insert a block comment in an open block, proceed as follows:

1. Click the small arrow in front of the block title.

   The right arrow becomes a down arrow, and the comment area is displayed.
2. Click "Comment" in the comment area.

   The "Comment" text passage is selected.
3. Enter the block comment.

Enter block comments for closed blocks

To insert the block comment in a closed block, follow these steps:

1. Right-click the block in the project tree.
2. Select the "Properties" command in the shortcut menu.
   The dialog with the properties of the block opens.
3. Select the entry "Information" in the area navigation.
4. Enter the block comment in the "Comment" input field.
5. Confirm your entry with "OK".

See also

Creating organization blocks (Page 844)

Creating functions and function blocks (Page 845)

Entering a block title (Page 851)

9.1.3.2 Specifying block properties

Basics of block properties

Block properties

Each block has certain properties, which you can display and edit. These properties are used amongst other things to:

- Identify the block
- Display the memory requirements and the compilation status of the block
- Display the time stamp
- Display the reference information
- Specify the access protection

See also

Basics of block access (Page 700)

Overview of block properties (Page 854)

Block time stamps (Page 856)

Displaying and editing block properties (Page 858)

Setting the mnemonics (Page 895)

Overview of block properties

Overview

The properties of the blocks are block and CPU-specific. Not all properties are therefore available for all blocks or in all CPU families. The following table gives an overview of block properties:

Group	Property	Description
General	Name	Unique block name within the station
	Constant name	Name of the constant pasted for the OB in the PLC tag table
	Type	Block type (cannot be changed)
	Number	Block number
	Event class	Event class of an OB (cannot be changed)
	Language	Programming language of the block
	Language in networks	Language used to program conditions in GRAPH blocks.
Information	Title	Block title
	Comment	Block comment
	Version	Version number of the block
	Family	Block family name
	Author	Name of the author, company name, department name, or other names
	User-defined ID	ID created by the user
Time stamps	Block	Times of creation and time of change of the block (cannot be changed)
	Interface	Time of creation of the block interface (cannot be changed)
	Code	Code modification time (non-editable)
	Data	Data modification time (non-editable)
Compilation	Status	Details of the last compilation run (cannot be changed)
	Lengths	Details of the block lengths (cannot be changed)
Protection	Protection	Setting up know-how protection and copy protection for the block See also: Protecting blocks
Attributes	Optimized block access	The tag declaration for blocks with optimized access contains only the symbolic names of the data elements. The addresses are automatically optimized and managed by the system. The performance of the CPU is improved, and access errors such as those from SIMATIC-HMI cannot occur. See also: Basics of block access (Page 700)
	IEC Check	The compatibility of the operands in comparison operations and arithmetic operations are tested according to IEC 61131. You have to explicitly convert non-compatible operands. See also: Overview of data type conversion (Page 780)
	Handle errors within block	Error handling inside the block with the GetError or GetErrorID instruction (cannot be changed). See also: Handling program execution errors (Page 1009)

Group	Property	Description
	Create extended status information	Allows you to monitor all tags in an SCL block. This option does, however, increase the program memory space required and the execution times.
	Check ARRAY limits	Checks whether field indexes are within the range declared for an ARRAY during the runtime of an SCL block. The block enable output ENO is set to "0" if a field index is outside of the permitted range.
	Set ENO automatically	Checks whether errors occur in the processing of certain instructions during the runtime of an SCL block. The block enable output ENO is set to "0" if a runtime error occurs.
	Create minimized DB	Generates instance data blocks for GRAPH data blocks in minimized format. This option reduces the GRAPH FB memory space required, however you will only receive limited program status information.
	Skip steps	If the transitions before and after a step become valid simultaneously in a GRAPH block, the step will not be activated and will be skipped.
	Acknowledgment required for supervision errors	Any supervision error which occurs during the operation of a GRAPH block must be acknowledged before the program can continue.
	Permanent processing of all interlocks in manual mode	Permanently monitors all interlock conditions in a GRAPH block in manual mode.
	Lock operating mode selection	Prevents a GRAPH block operating mode being selected.
	Data block write-protected in the device	Indicates whether the data block is read-only in the target system, and cannot be overwritten while the program is running (for data blocks only)
	Only store in load memory	On activation the data block is stored only in the load memory, occupies no space in the work memory, and is not linked into the program. The "Instructions" task card in the "Move" section offers options for the transfer of data blocks to the work memory. (only for data blocks)
Triggers	Triggers	Assigns the organization block to event by means of which it can be started. (only hardware interrupt OB) See also: Assigning parameters to hardware interrupt OBs
Cyclic interrupt	Cyclic interrupt	Settings for the cyclic interrupt OB See also: Assigning parameters to cyclic interrupt OBs

See also

Basics of block properties (Page 853)

Block time stamps (Page 856)

Displaying and editing block properties (Page 858)

Block time stamps

Introduction

Blocks receive a number of different time stamps which show you when the block was created and when it was last changed. These time stamps are also used for automatic consistency checks before compilation.

Code block time stamps

The following time stamps are generated for code blocks (OBs, FBs, FCs):

- Block: Created on, Modified on
- Interface: Modified on
- Code/data: Modified on

A time stamp conflict is displayed during compilation if the time stamp for the block calling is older than that of the interface for the block called.

Time stamps for code blocks are updated as follows:

- Block: The time stamp for the last block modification is always the same as the time stamp either of the interface or of the code depending on which area was modified last.
- Interface: The interface time stamp is updated each time the interface is modified. Even if you manually undo a change to the interface, for example change the name back, that is also a change which updates the time stamp. However, if you undo the change using the "Undo" function, the time stamp will be reset to the value it had before the undone change.
- Code/data: The code time stamp is updated each time the block code is changed. Even if you manually undo a change to the code, for example remove an instruction, that is also a change which will update the time stamp. However, if you undo the change using the "Undo" function, the time stamp will be reset to the value it had before the undone change.

Global data block time stamps

The following time stamps are generated for global data blocks:

- Block: Created on, Modified on
- Interface: Modified on
- Data: Modified on

Time stamp conflict is reported during compilation of a global data block based on a PLC data type if the time stamp of the global data block is older than the time stamp of the PLC data type used.

The time stamps for global data blocks are updated as follows:

- Block: The time stamp for the last change to a global data block is always the same as the time stamp for the interface and the data.
- Interface and data: The interface time stamps and the data are updated each time the global data block is changed. Even if you manually undo a change, for example remove a tag, that is also a change which will update the time stamp. However, if you undo the change using the "Undo" function, the time stamps will be reset to the value they had before the undone change.

Instance data block time stamps

The following time stamps are generated for instance data blocks:

- Block: Created on, Modified on
- Interface: Modified on
- Data: Modified on

A time stamp conflict will be reported during compilation of an instance data block if the interface time stamps of the instance data block are not identical to those of the function block.

The time stamps for instance data blocks are updated as follows:

- Block: The time stamp for the last change to an instance data block is always the same as the time stamp for the interface and the data.
- Interface and data: The interface time stamps and the data are updated each time the instance data block is changed. Even if you manually undo a change, for example cancel the tag retain setting, that is also a change which will update the time stamps. However, if you undo the change using the "Undo" function, the time stamps will be reset to the value they had before the undone change.

PLC data type time stamps

The following time stamps are generated for PLC data types:

- Block: Created on, Modified on
- Interface: Modified on

The time stamps for PLC data types are updated as follows:

- Block: The time stamp for the last change to a PLC data type is always the same as the interface time stamp.
- Interface: The interface time stamp is updated each time the PLC data type is changed. Even if you manually undo a change, for example delete the content of a PLC data type, that is also a change which will update the time stamp. However, if you undo the change using the "Undo" function, the time stamp will be reset to the value it had before the undone change.

See also

Basics of block properties (Page 853)

Overview of block properties (Page 854)

Displaying and editing block properties (Page 858)

Basic information on compiling blocks (Page 1071)

Displaying and editing block properties

The properties of the blocks are block and CPU-specific. Not all properties are therefore available for all blocks or in all CPU families. Properties that can only be displayed are write-protected.

Displaying and editing properties of a closed block

To display and edit the properties of a closed block, follow these steps:

1. Open the "Program blocks" folder in the project tree.
2. Right-click the block whose properties you want to display or edit.
3. Select the "Properties" command in the shortcut menu.

   The properties dialog box of the block opens.
4. In the area navigation, click a group whose properties you want to display or edit.
5. Change the relevant property.
6. Confirm your entries with "OK".

Displaying and editing properties of an open block

To display or edit the properties of an open block, follow these steps:

1. Select the "Inspector window" check box in the "View" menu.

   The Inspector window opens.
2. Click the "Properties" tab.

   The properties of the block are shown in the "Properties" tab of the Inspector window.
3. In the area navigation, click a group whose properties you want to display or edit.
4. Change the relevant property.

Result

The properties of the block will be changed. The changes are not saved until the project is saved.

See also

Basics of block properties (Page 853)

Overview of block properties (Page 854)

Block time stamps (Page 856)

9.1.3.3 Managing blocks

Opening blocks

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To open a block, follow these steps:

1. Double-click on the block you wish to open.

Result

The block will open in the program editor.

See also

Saving blocks (Page 860)

Closing blocks (Page 860)

Renaming blocks (Page 861)

Deleting blocks offline (Page 861)

Deleting blocks online (Page 862)

Opening know-how protected blocks (Page 1089)

Saving blocks

Blocks are always saved together with the project. Faulty blocks can also be saved. This allows the error to be resolved at a convenient time.

Procedure

To save a block, follow these steps:

1. Select the "Save" or "Save as" command in the "Project" menu.

   See also: Saving projects (Page 213)

See also

Opening blocks (Page 859)

Closing blocks (Page 860)

Renaming blocks (Page 861)

Deleting blocks offline (Page 861)

Deleting blocks online (Page 862)

Closing blocks

Procedure

To close a block, follow these steps:

1. Click the "Close" button in the title bar of the program editor.

NOTICE
Note that the block will not be saved on closing.

See also

Opening blocks (Page 859)

Saving blocks (Page 860)

Renaming blocks (Page 861)

Deleting blocks offline (Page 861)

Deleting blocks online (Page 862)

Renaming blocks

Requirement

The "Program blocks" folder is opened in the project tree.

Procedure

To change the name of a block, follow these steps:

1. Right-click the block that you want to rename.
2. Select the "Rename" command in the shortcut menu.

   The block name in the project tree changes to an input field.
3. Input the new name for the block.
4. Confirm your entry with the Enter key.

Result

The name of the block is now changed at all points of use in the program.

See also

Opening blocks (Page 859)

Saving blocks (Page 860)

Closing blocks (Page 860)

Deleting blocks offline (Page 861)

Deleting blocks online (Page 862)

Deleting blocks offline

Requirement

The "Program blocks" folder opens in the project tree.

Procedure

To delete a block that exists offline, proceed as follows:

1. In the project tree in the "Program blocks" folder, right-click on the block that you want to delete.
2. Select the "Delete" command in the shortcut menu.
3. Confirm the safety prompt with "Yes".

   The block is deleted offline from the project.

Note

If you are deleting organization blocks, note that events may be assigned to these blocks. If you delete such organization block the program cannot respond to parameterized events.

See also

Opening blocks (Page 859)

Saving blocks (Page 860)

Closing blocks (Page 860)

Renaming blocks (Page 861)

Deleting blocks online (Page 862)

Deleting blocks online

Note

S7-1200 Version 1.0

If you delete online blocks, the CPU will execute a cold restart the next time it switches to RUN mode. Apart from deleting the inputs, initializing the outputs and deleting the non-retentive memory, cold restart also deletes the retentive memory areas. All subsequent changes from STOP to RUN are warm restarts in which the retentive memory is not deleted.

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To delete a block that exists online, follow these steps:

1. In the "Program blocks" folder in the project tree, right-click on the block that you wish to delete from the device.
2. Select the "Delete" command in the shortcut menu.

   The "Delete" dialog will open.
3. Select the "Delete from device" option button.
4. Click on "Yes".

   The block will be deleted from the device online.

See also

Opening blocks (Page 859)

Saving blocks (Page 860)

Closing blocks (Page 860)

Renaming blocks (Page 861)

Deleting blocks offline (Page 861)

9.1.4 Programming blocks

9.1.4.1 Program editor

Overview of the program editor

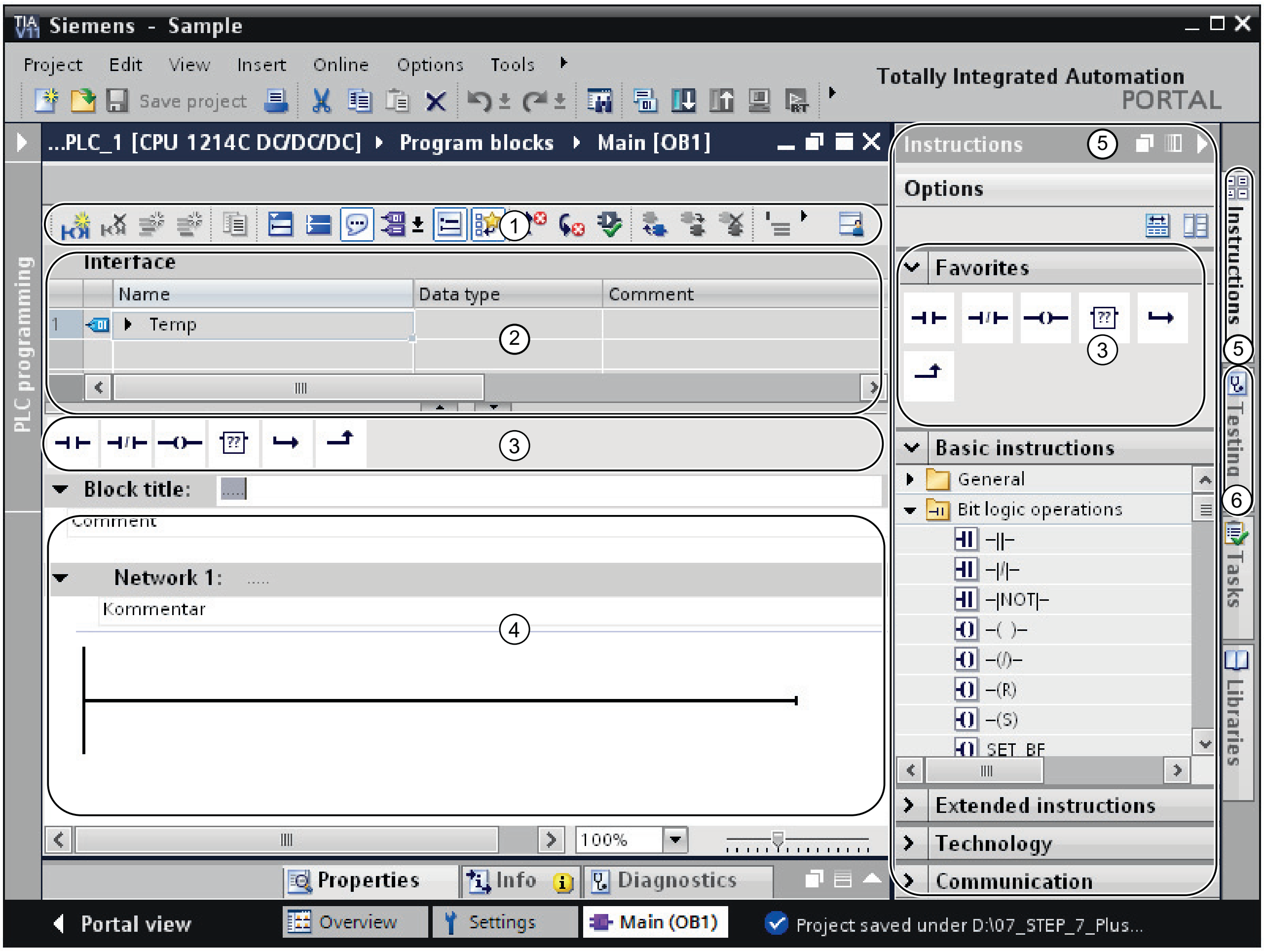
Function of the program editor

The program editor is the integrated development environment for programming functions, function blocks, and organization blocks. If offers comprehensive support for programming and troubleshooting.

The appearance and functionality of the program editor can vary depending on the programming language and the block type used.

Structure of the program editor

Using LAD as an example, the following figure shows the components of the program editor:



① Toolbar
② Block interface
③ "Favorites" pane in the "Instructions" task card and favorites in the program editor
④ Programming window
⑤ "Instructions" task card
⑥ "Testing" task card

Toolbar

The toolbar allows you to access the principal functions of the program editor, such as:

- Show and hide absolute operands
- Show and hide favorites
- Skip to syntax errors
- Update block calls
- Show and hide program status

The functions available in the toolbar can vary depending on the programming language used.

Block interface

The block interface contains the declarations for local tags used solely within the block. The sections available depend on the block type.

Favorites

You can save frequently used instructions as favorites. These favorites are then displayed in the "Instructions" task card and the "Favorites" pane. You can also display favorites in the program editor using the program editor toolbar. This allows you to access your favorites even when the "Instructions" task card is not visible.

Programming window

The programming window is the work area of the program editor. You can enter the program code in this window. The appearance and functionality of the program window can vary depending on the programming language used.

"Instructions" task card

The "Instructions" task card offers you easy access to all instructions available for creating your program. The instructions are broken down by area into a number of different palettes. You can show additional information on the instructions via the "Show column headers and additional columns" button in the task card toolbar.

"Testing" task card

You can set settings in the "Testing" task card for troubleshooting using the program status. The functions of the task card "Testing" are available only in online mode. It contains the following panes which are shown depending on the programming language set for the block:

- CPU operator panel

  The CPU operator panel allows you to switch the CPU operating mode.
- Breakpoints

  You can test blocks you created in the textual programming languages STL and SCL in single step mode. To do this, set breakpoints in the program code.

  You can find, activate, delete, navigate to, or set the call environment for the individual breakpoints you have set in the pane "Breakpoints".
- PLC register

  This pane allows you to read off the values for the PLC registers and the accumulators.
- Sequence control

  In this pane you can set the operating mode for testing sequencers for GRAPH blocks.
- Test settings

  This pane allows you to specify the test settings for the GRAPH block.
- Call environment

  This pane allows you to specify the call environment for the block.
- Call hierarchy

  In this pane, you can trace the call hierarchy of the blocks. You only see the call hierarchy during block monitoring.

See also

Layout of the block interface (Page 876)

Enlarging the programming window (Page 869)

Using the keyboard in the program editor

Navigate in the editor

Function	Keyboard shortcut
Open "Instructions" task card	<Ctrl+Shift+C>
Open "Testing" task card	<Ctrl+Shift+O>

Navigating in the program code (LAD/FBD)

Function	Selected object	Keyboard shortcut
Navigate between objects in the network.	Object in the network	Arrow keys

Navigating in the program code (STL/SCL)

Function	Position of the cursor	Keyboard shortcut
Navigate the program code.	Line	Arrow keys
One word to the right/left	Line	<Ctrl+arrow keys>
Cursor to start of line	Line	<Home>
Cursor to end of line	Line	<End>
Cursor to start of code section	Line	<Ctrl+Home>
Cursor to end of code section	Line	<Ctrl+End>
To the next network	Network title	<Down arrow>
To the next network	Line	<Ctrl+Tab>
To the previous network	Network title	<Up arrow>
To the previous network	Line	<Ctrl+Shift+Tab>

Inserting instructions (LAD)

Function	Selected object	Keyboard shortcut
Insert normally open contact	Rung	<Shift+F3>
Insert normally closed contact	Rung	<Shift+F4>
Insert empty box	Rung	<Shift+F5>
Insert assignment	Rung	<Shift+F7>
Insert "Open nesting"	Rung	<Shift+F9>
Insert "Close branch"	Rung	<Shift+F11>

Inserting instructions (FBD)

Function	Selected object	Keyboard shortcut
Insert assignment	Insert network input or output	<Shift+F7>
Insert empty box	Network	<Shift+F5>
Insert "Open nesting"	Connection line between two boxes	<Shift+F9>
Invert RLO	Insert network input or output	<Alt+4>
Insert input	Insert network input or output	<Alt+3>

Enter operands (LAD/FBD)

Function	Selected object	Keyboard shortcut
Enable the input field for the first operand of the instruction.	Instruction	<Enter> Or <Any letters/numbers> An empty input field opens on <Return>, any letters or numbers will be entered in the entry field.
Enable input field for the operand.	Operand	<F2>
Delete value of the operand.	Operand	<Del>
Define tag	Operand	<Alt+Shift+D>
Entering operands	Input field for operands	<Any letters/numbers>
Confirm entry of the operand.	Input field for operands	<Enter>
Open autocompletion.	Input field for operands	<Ctrl+I>
Discard current change.	Input field for operands	<Esc> The input field is deactivated and the previous content restored.

Process instructions (STL/SCL)

Function	Selected object	Keyboard shortcut
Indent line	Line	<Tab>
Outdent line	Line	<Shift+Tab>
Open "Call options" dialog	Cursor before block call	<Enter>
Define tag	Operand	<Alt+Shift+D>
Open autocompletion.	Any	<Ctrl+I> or <Ctrl+Spacebar>
Set/delete bookmarks		<Ctrl+Alt+B>
To the next bookmark		<Alt+Shift+7>
To the previous bookmark		<Alt+Shift+6>

Monitor program

Function	Keyboard shortcut
Monitor from here	<F5>
Set/delete breakpoint	<F9>
Skip	<F10>
Jump to	<F11>
Execution to selection	<Shift+F5>
Display program status	<Shift+T>
Activate all breakpoints	<Ctrl+Shift+F9>
Deactivate all breakpoints	<Ctrl+Shift+F10>

Enlarging the programming window

Introduction

The programming window is relatively small when all components of the application are shown. If the program code is large, you may find you have to rearrange the work area constantly. To avoid this problem, you can hide or minimize the display of the following components of the application and of the program editor:

- Project tree
- Task cards
- Block interface
- Favorites
- Comments
- Networks

Note

You can also use the "Reduce automatically" option for the task cards, project tree, and Inspector window. These windows will then be minimized automatically when you do not need them.

See also: Maximizing and minimizing the work area

Hiding and showing the project tree

The project tree allows you to access all areas of the project. You can hide the project tree while you are creating a program so you have more space for the programming window.

To show and hide the project tree, follow these steps:

1. To hide the project tree, deselect the "Project tree" check box in the "View" menu, or click on "Collapse" on the project tree title bar.
2. To show the project tree, select the "Project tree" check box in the "View" menu or click on "Extend" on the project tree title bar.

Opening and closing task cards

The task cards are located at the right-hand edge of the programming window.

To open or close the task cards, follow these steps:

1. To close the task cards, deselect the "Task card" check box in the "View" menu or click "Collapse" on the task cards title bar.
2. To open the task cards, select the "Task card" check box in the "View" menu or click "Expand" on the task cards title bar.

Hiding and showing the block interface

The block interface is shown in the upper section of the program editor. During programming you can show and hide it as required.

To show and hide the block interface, follow these steps:

1. In the lower part of the interface within the window splitter, click on the Up arrow or Down arrow.

Showing and hiding favorites

To hide or show the favorites in the program editor, follow these steps:

1. Click the "Display favorites in the editor" button in the program editor toolbar.

Showing and hiding comments

Within a block you can enter a comment for the block or for each network. These two types of comments are shown and hidden differently.

To show or hide a block comment, follow these steps:

1. Click the the triangle at the start of the line with the block title.

To show or hide network comments, follow these steps:

1. Click "Network comments on/off" on the program editor toolbar.

Note

The comments available can vary depending on the programming language used.

Opening and closing networks

Some programming languages use networks. You can open or close these networks as required.

To open or close networks, follow these steps:

1. If you want to open a network, click the right arrow in front of the network title. If you want to close a network, click the down arrow in front of the network title.

To open and close all networks, follow these steps:

1. Click "Open all networks" or "Close all networks" in the program editor toolbar.

> **Note**
>
> Networks are not used in every programming language.

See also

Overview of the program editor (Page 863)

Using autocompletion

Basics of autocompletion

Function

You can use autocompletion in the program window of the program editor as an easy way to access available tags or instructions during programming. Autocompletion means a context-specific list appears in a dialog from which you can select the tags or instructions you need.

See also

Using autocompletion in graphic programming languages (Page 872)

Using autocompletion in textual programming languages (Page 873)

Using autocompletion in graphic programming languages

Inserting tags using autocompletion

To insert tags in graphic programming languages using autocompletion, follow these steps:

1. Select an operand of the instruction to which you wish to assign a tag.

   The input field for the operand will open. The autocompletion button will appear beside the input field.

2. Either click on the autocompletion button or type in the shortcut <Ctrl+I>.

   Autocompletion will open. It will contain only those local and global tags, data blocks and multi-instances which are admissible for the operand in the given context. You can exit autocompletion at any time by pressing <Esc>.

3. Select the required tag from the list. If necessary, you can also filter the list:

   – For example, enter the first few letters of the name of the tag or instruction you wish to insert. Autocompletion will be filtered further with each letter entered. If there is no tag or instruction starting with the letters entered, autocompletion will remain at the last match.

   – Enter # to access the local tags from the block interface.

   – Enter " to access the global tags.

   If the tag is a structured tag, a data block or a multi-instance, then an arrow will be displayed at the end of the row. Click on the arrow to display the subordinate element. You can navigate to the very last level in this way. Use the <Backspace> key to return to the previous level.

4. Press the <Return> key to apply the tag.

See also

Basics of autocompletion (Page 871)

Using autocompletion in textual programming languages (Page 873)

Using autocompletion in textual programming languages

Inserting tags and instructions using autocompletion

To insert tags and instructions in textual programming languages using autocompletion, follow these steps:

1. Enter the first few letters of the name of the tag or instruction you wish to insert. If necessary, you can directly filter the kind of tags:
   - Enter # to access the local tags from the block interface.
   - Enter " to access the global tags.

   Autocompletion will open. It will contain only those local and global tags, data blocks, multi-instances and instructions which are admissible in the current position. You can exit autocompletion at any time by pressing <Esc>.
2. Enter more letters of the name of the tag or instruction you wish to insert. Press the <Space bar> to close autocompletion.

   Autocompletion will be filtered further with each letter entered. If there is no tag or instruction starting with the letters entered, autocompletion will remain at the last match.
3. Select the tag or instruction required from the list.

   If the tag is a structured tag, a data block or a multi-instance, then an arrow will be displayed at the end of the row. Click on the arrow to display the subordinate element. You can navigate to the very last level in this way. Use the <Backspace> key to return to the previous level.
4. Press the <Return> key to apply the tag.

See also

Basics of autocompletion (Page 871)

Using autocompletion in graphic programming languages (Page 872)

General settings for the PLC programming

Overview of the general settings

Overview

The following table shows the general settings that you can make:

Group	Setting	Description
View	With comments	Network comments are shown.
Compilation	Delete actual parameters on interface update	Actual parameters are deleted if the associated formal parameters were deleted in the called block, and you run the "Update block call" function or compile the block.
Default settings for new blocks	IEC check	The compatibility of operands in comparison operations and arithmetic operations are tested according to IEC rules. You have to explicitly convert non-compatible operands.
	Optimized block access (S7-1200)	The tag declaration for blocks with optimized access contains only the symbolic names of the data elements. The addresses are automatically optimized and managed by the system. The performance of the CPU is improved, and access errors such as those from SIMATIC-HMI cannot occur.
Additional settings	Show autocomplete list	The autocomplete list is displayed.
	Mnemonics	German or international designation of operations and operands

See also

Changing the settings (Page 875)

Permissible addresses and data types of PLC tags (Page 829)

Basics of block access (Page 700)

Setting and canceling the IEC check (Page 782)

Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.

   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the general settings (Page 874)

9.1.4.2 Programming code blocks

Declaring the block interface

Basic information on declaring the block interface

Introduction

The interface contains the declarations of local tags that are used within the block. The tags are subdivided into two groups:

- Block parameters that form the block interface when it is called in the program.
- Local data that are used for storage of intermediate results.

Purpose of tag declaration

You use tag declaration to define the call interface of a block in the program and the names and data types of tags that you want to use in the block.

The interface of function blocks also defines the structure of the instances that are assigned to the function block.

Using POINTER

The following rules apply to declaration of the interface of a function:

- Local tags declared in the section "Input" of the interface cannot be used for the outputs of the programmed instructions or called blocks.
- Local tags declared in the section "Output" of the interface cannot be used for the inputs of the programmed instructions or called blocks.
- Local tags declared in the section "InOut" of the interface can be used for the inputs and outputs of the programmed instructions or called blocks.

The following rules apply to the use of block parameters within a function block (FB):

- Local tags declared in the section "Input" of the interface cannot be used for the outputs of the programmed instructions or called blocks if the IEC check is enabled. Use is possible if the IEC check is disabled.
- Local tags declared in the section "Output" of the interface cannot be used for the inputs of the programmed instructions or called blocks if the IEC check is enabled. Use is possible if the IEC check is disabled.
- Local tags declared in the section "InOut" of the interface can be used for the inputs and outputs of the programmed instructions or called blocks regardless of the IEC check settings.

See also

Using tags within the program (Page 717)

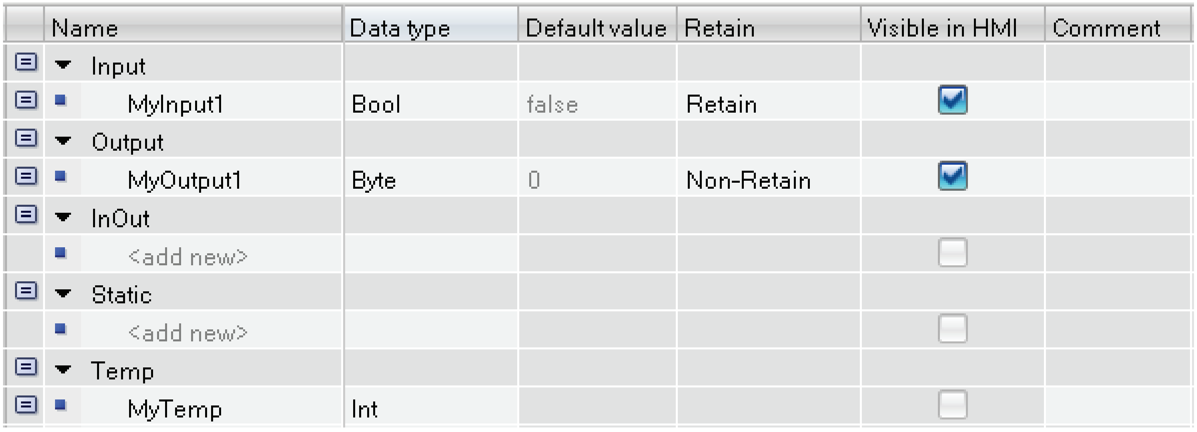
Reserved key words (Page 719)

Layout of the block interface

The block interface allows local tags to be created and managed.

Layout of the block interface

The following figure shows the structure of the block interface. The number of columns and sections varies depending on the type of block.



Name	Data type	Default value	Retain	Visible in HMI	Comment
Input					
MyInput1	Bool	false	Retain	☑	
Output					
MyOutput1	Byte	0	Non-Retain	☑	
InOut					
<add new>				□	
Static					
<add new>				□	
Temp					
MyTemp	Int			□	

Block parameters

The following table shows the types of block parameters:

Type	Section	Function	Available in
Input parameters	Input	Parameters whose values are read by the block.	Functions, function blocks and some types of organization blocks
Output parameters	Output	Parameters whose values are written by the block.	Functions and function blocks
In/out parameters	InOut	Parameters whose values are read by the block when it is called, and whose values are written again by the block after execution.	Functions and function blocks
Return value	Return	Function value that is returned to the calling block.	Functions

Depending on the type of block opened, additional sections may be displayed.

Local data

The following table shows the types of local data:

Type	Section	Function	Available in
Temporary local data	Temp	Tags that are used to store temporary intermediate results. Temporary local data are retained for only one cycle. If you use temporary local data, you have to make sure that the values are written within the cycle in which you want to read them. Otherwise the values will be random.	Functions, function blocks and organization blocks
Static local data	Static	Tags that are used for storage of static intermediate results in the instance data block. Static data is retained until overwritten, which may be after several cycles. The names of the blocks, which are called in this code block as multi-instance, will also be stored in the static local data.	Function blocks

Meaning of the columns

The following table shows the meaning of the individual columns: You can show or hide the columns as required.

Column	Explanation
🏷	Symbol you can click on to drag-and-drop a tag to a program for use as an operand.
Name	Name of the tags.
Data type	Data type of the tags.
Default value	Value with which you pre-assign the tag in the interface of the code block. Specification of the default value is optional. If you do not specify any value the predefined value for the indicated data type is used. For example, the value "false" is predefined for BOOL. The default value is assumed as the start value in the corresponding instance data block. You can replace these values with instance-specific start values in the instance data block. The column is only available in the interface of function blocks.
Retain	Marks the tag as retentive. The values of retentive tags are retained even after the power supply is switched off. This column is only visible in the interface of the function block with optimized access.
Visible in HMI	Shows whether the tag is visible by default in the HMI selection list.
Accessible from HMI	Shows whether HMI can access this tag during runtime. The column is only visible on S7-1200-CPUs.
Comment	Comments to document the tags.

See also

Using tags within the program (Page 717)

Reserved key words (Page 719)

Valid data types in the code block interfaces (Page 879)

Setting the retentivity of local tags (Page 891)

Valid data types in the code block interfaces

Valid data types for the interface of organization blocks

The following table shows the valid data types for the interface of an organization block in S7-1200:

	S7-1200		
Section	Elementary Data types	Structured Data types	VARIANT
Temp	X	X	-

The following table shows the valid data types for the interface of an organization block in S7-300/400:

	S7-300/400				
Section	Elementary Data types	Structured Data types	TIMER COUNTER BLOCK	POINTER	ANY
Temp	X	X	-	-	X

Valid data types for the interface of function blocks

The following table shows the valid data types for the interface of a function block in S7-1200:

	S7-1200		
Section	Elementary Data types	Structured Data types	VARIANT
Input	X	X	X
Output	X	X	-
InOut	X	X[(1)]	x
Static	X	X	-
Temp	X	X	-
[(1)] STRING can only be defined in the standard length of 254 characters.			

The following table shows the valid data types for the interface of a function block in S7-300/400:

Section	S7-300/400				
	Elementary Data types	Structured Data types	TIMER COUNTER BLOCK	POINTER	ANY
Input	X	X	X	X	X
Output	X	X	-	-	-
InOut	X	X[(1)]	-	X	X
Static	X	X	-	-	-
Temp	X	X	-	-	X
[(1)] STRING can only be defined in the standard length of 254 characters.					

Valid data types for the interface of a function

The following table shows the valid data types for the interface of a function block in S7-1200:

Section	S7-1200		
	Elementary Data types	Structured Data types	VARIANT
Input	X	X[(1)]	X
Output	X	X[(1)]	X
InOut	X	X[(1)]	X
Temp	X	X	-
[(1)] STRING can only be defined in the standard length of 254 characters.			

The following table shows the valid data types for the interface of a function block in S7-300/400:

Section	S7-300/400				
	Elementary Data types	Structured Data types	TIMER COUNTER BLOCK	POINTER	ANY
Input	X	X[(1)]	X	X	X
Output	X	X[(1)]	-	X	X
InOut	X	X[(1)]	-	X	X
Temp	X	X	-	-	X
[(1)] STRING can only be defined in the standard length of 254 characters.					

Declaring local tags

Declaring local tags in the block interface

Requirement

The block interface is open.

Procedure

To declare a tag of the elementary data type, follow these steps:

1. Select the appropriate declaration section in the interface:
2. Enter a tag name in the "Name" column.
3. Enter the required data type in the "Data type" column. You will be supported by autocompletion during input.
4. Optional: Change the properties of the tags that are displayed in the other columns of the block interface.

Result

The tag is created.

Syntax check

A syntax check is performed after each entry, and any errors found are displayed in red. You do not have to correct these errors immediately - you can continue editing and make any corrections later. However, you will not be able to compile the program if the tag declaration contains syntax errors.

Note

If you change the interface of a block, the calls of the block in the program will possibly become inconsistent. The call locations are automatically updated, if possible.

If an automatic updating is not possible, the inconsistent blocks have to be updated manually.

See also:

Updating block calls in LAD (Page 918)

Updating block calls in FBD (Page 956)

See also

Basic information on start values (Page 1023)

Using tags within the program (Page 717)

Reserved key words (Page 719)

Valid data types in the code block interfaces (Page 879)

Properties of local tags (Page 890)

Setting the retentivity of local tags (Page 891)

Declaring local tags in the program editor

Requirement

The program editor is open.

Procedure

To declare a local tag, follow these steps:

1. Insert an instruction in your program.

   The "<???>", "<??.?>" or "..." strings represent operand placeholders.
2. Replace an operand placeholder with the name of the tag to be created.
3. Select the tag name.

   If you want to declare multiple tags, select the names of all the tags to be declared.
4. Select the "Define tag" command in the shortcut menu.

   The "Define tag" dialog box opens. This dialog displays a declaration table in which the name of the tag is already entered.
5. To declare a local tag, select one of the following sections:
   - Local In
   - Local Out
   - Local InOut
   - Local Static
   - Local Temp
6. In the other columns, enter data type and comments.
7. Click the "Define" button to complete your entry.

Result

The declaration is written directly into the block interface and is valid within the entire block.

Note

If you change the interface of a block, the calls of the block in the program will possibly become inconsistent. The call locations are automatically updated, if possible.

If an automatic updating is not possible, the inconsistent blocks have to be updated manually.

See also:

Updating block calls in LAD (Page 918)

Updating block calls in FBD (Page 956)

See also

Using tags within the program (Page 717)

Reserved key words (Page 719)

Valid data types in the code block interfaces (Page 879)

Basic information on start values (Page 1023)

Properties of local tags (Page 890)

Setting the retentivity of local tags (Page 891)

Declaring tags of the ARRAY data type

Requirement

The block interface is open.

Procedure

To declare a tag of the ARRAY data type, follow these steps:

1. Select the appropriate declaration section in the interface.
2. Enter a tag name in the "Name" column.
3. In the "Data type" column, click the button for the data type selection.

   A list of the permissible data types is opened.
4. Select the "Array" data type.

   The "Array" dialog opens.
5. In the "Data type" text box, specify the data type of the array elements.
6. In the "ARRAY limits" text box, specify the high and low limit for each dimension.

   Example of a one-dimensional ARRAY:

   ```
   [0..3]
   ```

   Example of a three-dimensional ARRAY:

   ```
   [0..3, 0..15, 0..33]
   ```
7. Confirm your entry.
8. Optional: Change the properties of the tags that are displayed in the other columns of the block interface.

Result

The tag of ARRAY data type is created.

Note

You cannot define specific default values for ARRAY elements. However, you can assign them start values in the instance.

See also

Using tags within the program (Page 717)

Reserved key words (Page 719)

Properties of local tags (Page 890)

Setting the retentivity of local tags (Page 891)

Declaring tags of STRUCT data type

Requirement

The block interface is open.

Procedure

To declare a tag of the STRUCT data type, follow these steps:

1. Select the appropriate declaration section in the interface:
2. Enter a tag name in the "Name" column.
3. Enter "Struct" in the "Data type" column. You will be supported by autocompletion during input.

   An empty, indented row is inserted after the new tag.
4. Insert the first structural element in the first empty row.

   An additional empty row is inserted after the element.
5. Select a data type for the structure element.
6. Optional: Change the properties of the structural element that is displayed in the other columns of the block interface.
7. Repeat the step 4 to 7 for all additional structure elements.

   It is not necessary to end the structure explicitly. The structure ends with the last element that is entered.
8. To insert a new tag after the structure, leave a blank row after the end of the structure and then start the new tag in the second empty row.

Result

The tag of STRUCT data type is created.

See also

Using tags within the program (Page 717)

Reserved key words (Page 719)

Properties of local tags (Page 890)

Setting the retentivity of local tags (Page 891)

Declaring tags based on a PLC data type

Requirement

A PLC data type is declared in the current CPU.

Procedure

To declare a tag based on a PLC data type, follow these steps:

1. Select the appropriate declaration section in the interface:
2. Enter the PLC data type in the "Data type" column. You will be supported by Autocomplete during input.

Result

The tag is created.

Note

You define the default values of tags within a PLC data type when the PLC data type is created. You cannot change these values at the point of use of the PLC data type.

If you change or delete PLC data types that are used in the block interface, the interface becomes inconsistent. To remedy this inconsistency, the interface has to be updated.

See also: Updating the block interface (Page 888)

See also

Basics of PLC data types (Page 1036)

Declaring higher-level tags

Introduction

To access data areas within a declared tag, you can overlay the declared tags with an additional declaration. This provides you with the option of addressing an already declared tag with a different data type. You can, for example, address the individual bits of a tag of WORD data type with an ARRAY of BOOL.

Overlaying tags

To overlay a tag with a new data type, follow these steps:

1. Open the block interface.
2. In the interface, select the tag that you want to overlay with a new data type.
3. Click "Add row" in the toolbar.

   A row is inserted after the tag to be overlaid. The overlaying tag must be declared in the row directly after the tag that is to be overlaid.
4. Enter a tag name in the "Name" column.
5. Enter the "AT" entry in the "Data type" column. You will be supported by Autocomplete in this step.

   The following is added to the entry in the "Name" column.

   ```
   "AT<Name of the higher-level tag>"
   ```
6. Click the data type selection button again and select the data type for the new tag.

   The tag is created. It points to the same data as the higher-level tag, however interprets this data with the new data type.

Removing overlay

To remove the overlay of a tag, follow these steps:

1. Select the overlaid tag that you want to remove.
2. Select the "Delete" command in the shortcut menu.
3. The overlay is removed.

See also

Overlaying tags with AT (Page 731)

Declaring multi-instances

Requirement

- The function block to be called exists in project tree and is multi-instance capable.
- The block interface of the calling function block is open.

Procedure

To declare a function block to be called as a multi-instance, follow these steps:

1. In the "Name" column of the "Static" section, enter a designation for the block call.
2. In the "Data type" column, enter the symbolic name for the function block to be called.

Note

The program editor will declare the multi-instance automatically if you program a block call in a network and then specify in the "Call options" dialog that you want to call the block as a multiple instance.

See also

Updating the block interface (Page 888)

Updating the block interface

Introduction

If you change or delete PLC data types or multiple instances that are used in the block interface, the interface will become inconsistent. To remedy this inconsistency, the interface has to be updated.

You have two options for updating the block interface:

- Explicit updating of the block interface.
  The used PLC data types and multiple instances will be updated. The instance data blocks that belong to the block are not implicitly updated during this process.
- Implicit updating during compilation.
  All used PLC data types and multiple instances as well as the related instance data blocks will be updated.

Explicit updating of the block interface

To explicitly update the block interface, follow these steps:

1. Open the block interface.
2. Select the "Update" command in the shortcut menu.

Implicit Updating during Compilation

Proceed as follows to implicitly update all uses of PLC data types and multiple instances as well as the instance data blocks during compilation:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Basics of PLC data types (Page 1036)

Declaring tags based on a PLC data type (Page 886)

Basic information on start values (Page 1023)

Using tags within the program (Page 717)

Reserved key words (Page 719)

Valid data types in the code block interfaces (Page 879)

Properties of local tags (Page 890)

Setting the retentivity of local tags (Page 891)

Updating block calls in LAD (Page 918)

Declaring multi-instances (Page 888)

Editing the properties of local tags

Properties of local tags

Properties

The following table gives an overview of the properties of local tags:

Group	Property	Description
General	Name	Name of the tags.
	Data type	Data type of the tags.
	Default value	Value with which you pre-assign the tag in the interface of the code block. Specification of the default value is optional. If you do not specify any value the predefined value for the indicated data type is used. For example, the value "false" is predefined for BOOL. The default value will be adopted as the start value in the corresponding instance. You can then replace these adopted values with instance-specific start values. This property is only available in the interface of function blocks.
	Comment	Comment on the tag.
Attributes	Retain	Marks the tag as retentive. The values of retentive tags are retained even after the power supply is switched off. This attribute is only available in the interface of the function block with optimized access.
	Accessible from HMI	Shows whether HMI can access this tag during runtime.
	Visible in HMI	Shows whether the tag is visible by default in the HMI selection list.
	Can be set	Indicates whether a parameter is configurable in CFC.
	For test	Indicates whether a parameter is registered for the CFC test mode.
	Visible	Indicates whether a parameter is visible in CFC.
	Interconnectable	Indicates whether a parameter is interconnectable in CFC.

See also

Setting the retentivity of local tags (Page 891)

Changing properties of local tags (Page 891)

Reserved key words (Page 719)

Setting the retentivity of local tags

Introduction

Function blocks store their data in an instance. To prevent data loss in the event of power failure, you can mark the data as retentive. This data is stored in a retentive memory area. The option of setting the retentivity depends on the set access type of the function block.

Retentive behavior in blocks with standard access

In blocks with standard access you cannot set the retentive behavior of individual tags. You can only define them as retentive in the assigned instance. All tags contained in the block are then considered as retentive.

Retentivity for optimized block access

In data blocks with optimized access you can define the retentive behavior of individual tags.

For structured data type tags, the retentivity setting always applies to the entire structure. You can make no individual retentivity setting for individual elements within the structure.

The following settings are available:

- Retentive

  The values of the tags or the structure are available even after a power failure.
- Non-retentive

  The values of the tags or the structure are lost in the event of a power failure.
- Set in IDB

  The retentivity can be set in the instance data block. The setting that is made in the instance data block than applies, however, centrally to all tags that are selected with "Set in IDB".

See also

Properties of local tags (Page 890)

Basics of block access (Page 700)

Changing properties of local tags

Editing properties in the block interface

To edit the properties of one or more tags, follow these steps:

1. Open the block interface.
2. Change the entries in the columns.

Editing properties in the properties window

To edit the properties of an individual tag, follow these steps:

1. Select a tag in the table.

   The tag properties will be displayed in the Inspector window.
2. Change the entries in the inspector window.

Renaming tags directly in the program editor

To rename one or more tags, follow these steps:

1. Select one or more tags in the program.
2. Select the "Rename tag" command in the shortcut menu.

   The "Rename tag" dialog opens. This dialog box displays a declaration table with the selected tags.
3. Change the entries in the "Name" column.
4. Confirm the input by clicking the "Change" button.

Editing the data type or comment in the program editor

Proceed as follows to edit the data type or tag comment in the program editor:

1. Select the tag name.
2. Select the "Rewire tag" command in the shortcut menu.

   The "Rewire tag" dialog will open. The dialog shows a declaration table.
3. Change the entry in the "Data type" or "Comment" columns.
4. Click the "Change" button to confirm the input.

Effect in the program

In the case of a change of the tag's name, data type or address, each location of use of the tag is automatically updated in the program.

Note

If you change the interface of a block, the program may become inconsistent. The inconsistencies are automatically updated, if possible.

If an automatic updating is not possible, the inconsistent calls are marked in red. You than have to manually updated the inconsistencies.

See also:

Updating block calls in LAD (Page 918)

Updating block calls in FBD (Page 956)

See also

Layout of the block interface (Page 876)

Properties of local tags (Page 890)

Setting the retentivity of local tags (Page 891)

Basic information on start values (Page 1023)

Using tags within the program (Page 717)

Reserved key words (Page 719)

Valid data types in the code block interfaces (Page 879)

Updating the block interface (Page 888)

Editing the block interface

Inserting table rows

Procedure

Proceed as follows to insert a row above the selected row:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button on the toolbar of the table.

Result

A new row is inserted above the selected row.

Inserting table rows

Procedure

Proceed as follows to insert a row below the selected row:

1. Select the row below which you want to insert a new row.
2. Click the "Add row" button on the table toolbar.

Result

A new empty row will be inserted below the selected row.

Deleting tags

Procedure

To delete a tag, follow these steps:

1. Select the row with the tag to be deleted. You can also select several rows by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last row.
2. Select the "Delete" command in the shortcut menu.

Automatically filling in successive cells

You can load the contents of one or several table cells into the cells below, automatically filling in the successive cells.

If you automatically fill in cells in the "Name" column, a consecutive number will be appended to each name. For example, "Motor" will become "Motor_1".

You can define individual or more cells as well as entire rows as source area.

If less rows exist in the open table than you want to fill, then you will first have to insert additional empty rows.

Requirement

- The table is open.
- Sufficient declaration rows are available.

Procedure

To automatically fill in successive cells, follow these steps:

1. Select the cells to be loaded.
2. Click the "Fill" symbol in the bottom right corner of the cell.

   The mouse pointer is transformed into a crosshair.
3. Keep the mouse button pressed and drag the mouse pointer downwards over the cells that you want to fill in automatically.
4. Release the mouse button.

   The cells are filled in automatically.
5. If entries are already present in the cells that are to be automatically filled in, a dialog appears. In this dialog you can indicate whether you want to overwrite the existing entries or insert new rows for the new tags.

Show and hide table columns

You can show or hide the columns in a table as needed.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. In the shortcut menu, select the "Show/hide columns" command.

   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.

Editing tags with external editors

To edit individual tags in external table editors, such as Excel, you can export or import these tags using copy and paste. However, you cannot copy structured tags to an editor.

Requirements

The block interface and an external editor are opened.

Procedure

To export individual tags to an external editor and import them again, follow these steps:

1. Select one or more tags.
2. Select "Copy" in the shortcut menu.
3. Switch to the external editor and paste the copied tags.
4. Edit the tags as required.
5. Copy the tags in the external editor.
6. Select the tags in the external editor.
7. Switch back to the block interface.
8. Select "Paste" in the shortcut menu.

Creating program code

Setting the mnemonics

You can program blocks using German or international mnemonics. If you open the TIA portal for the first the international mnemonics is set as default. You can change the mnemonics at any time.

Procedure

To set the mnemonics, follow these steps:

1. Select the "Settings" command in the "Options" menu.

   The "Settings" window is displayed in the work area.
2. Select the "General" group in the area navigation.
3. In the "General settings" group, select the mnemonics that you want to use.

   The mnemonics is changed in all blocks.

Displaying symbolic and absolute addresses

You have the following options for displaying operands in the program editor:

- Symbolic representation

  The symbolic operands are displayed in the program. The corresponding absolute addresses are shown in tooltips if you hold the mouse pointer over the operand.
- Absolute representation

  The absolute addresses are displayed in the program. The corresponding symbolic operands are displayed in tooltips.
- Symbolic and absolute representation

  Symbolic operands and absolute addresses are displayed in program.

Requirement

The program editor is open.

Procedure

To change the representation of the operands, follow these steps:

1. Click the "Absolute/symbolic operands" button in the program editor toolbar.

   Each time you click the button, the representation and the symbol on the button change.

Or:

1. Click the small arrow next to the "Absolute/symbolic operands" button in the program editor toolbar.

   A drop-down list is displayed.
2. Select the required representation from the drop-down list.

   The symbol on the button changes.

See also

Basic information about operands (Page 716)

Use instruction versions

Basic information on instruction versions

The instructions available to you for programming the user program are managed in system libraries. If a new version of a system library is installed by an update, the newer versions of the instructions of this system library may also be installed.

If there are several versions for an instruction, these are listed in the "Instructions" task card after the respective instruction. If the instruction versions are not shown, you can show them via the "Show column headers and additional columns" button in the toolbar of the "Instructions" task card. You can then select the versions of the instructions to be used in the program from the dropdown-list box of the "Version" column. If you do not select any versions, the most recent versions are used.

Note

Please note the following:

- You can only ever use the same version of an instruction within a device.
- If you change the version of an instruction that other instructions depend on, the versions of the dependent instructions are also changed.
- If you select a version for an instruction that can not be run on the CPU used, the instruction is shaded out. This means that you cannot use this instruction in this version with your CPU.

Changes in the versions

New versions can be main versions or secondary versions. New versions, such as 2.0 or 3.0, have more substantial changes to them. New main versions may therefore result in changes to the block interface. New secondary versions, such as 1.3 or 1.4, contain lesser changes or remedies to errors.

Using instruction versions

You can decide within a device which version of an instruction you want to use. If you select another version for an instruction, the new version is specified for all locations of use of this instruction within your program. These instructions are identified in the program by a red frame. You must then download your program to the device to use the new instruction version.

Creating LAD programs

Basic information on LAD

LAD programming language

Overview of the Ladder Logic (LAD) programming language

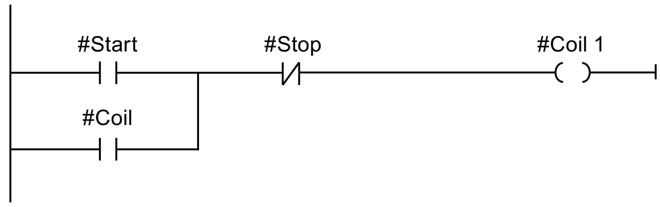
LAD is a graphical programming language. The representation is based on circuit diagrams.

The program is mapped in one or more networks. A network contains a power rail on the left where the rungs originate. The binary signal scans are arranged in the form of contacts on the rungs. The serial arrangement of the elements on a rung creates a series connection; arrangement on simultaneous branches creates a parallel connection. Complex functions are represented by boxes.

Example of networks in LAD

The following figure shows an LAD network with normally open contacts, normally closed contacts and a coil:

Network 1



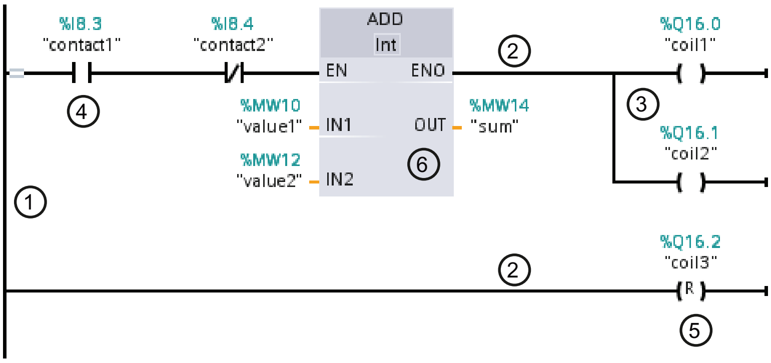
Overview of the LAD elements

LAD elements

A LAD program consists of separate elements that you can arrange in series or parallel on the power rail of a network. Most program elements must be supplied with tags.

There is at least one rung from the power rail. Network programming starts at the left edge of the rung. You can expand the power rail by several rungs and branches.

For example, the following figure shows elements of a LAD network:



1) Power rail

2) Rung

3) Branch

4) Contact

5) Coil

6) Box

Power rail

Each LAD network consists of a power rail that contains at least one rung. A network can be extended by adding additional rungs. You can use branches to program parallel connections in the specific rungs.

Contacts

You can use contacts to create or interrupt a current-carrying connection between two elements. The current is relayed from left to right. You can use contacts to query the signal state or the value of an operand and control it depending on the result of the current flow.

The following types of contact are available to you in a LAD program:

- Normally open contact:
  Normally open contacts forward the current if the signal state of a specified binary operand is "1".
- Normally closed contacts:
  Normally open contacts forward the current if the signal state of a specified binary operand is "0".
- Contact with additional function:
  Contacts with additional function forward the current if a specific condition is met. With these contacts you can also execute an additional function, such as an RLO edge detection and a comparison.

Coils

You can use coils to control binary operands. Coils can set or reset a binary operand depending on the signal state of the result of logic operation.

The following types of coils are available to you in a LAD program:

- Standard coils:
  Standard coils set a binary operand if current flows in the coil. The "Assignment" instruction is an example of a standard coil.
- Coils with additional function:
  These coils have additional functions besides the evaluation of the result of logic operation. Coils for RLO edge detection and program control are examples of coils with additional function.

Boxes

Boxes are LAD elements with complex functions. The empty box is an exception. You can use the empty box as a placeholder in which you can select the required instruction.

The following types of boxes are available to you in a LAD program:

- Boxes without EN/ENO mechanism:
  A box is executed depending on the signal state at the box inputs. The error status of the processing cannot be queried.
- Boxes with EN/ENO mechanism:
  A box is only executed if the enabling input "EN" carries the signal state "1". If the box is processed correctly, the "ENO" enable output has signal state "1". If an error occurs during the processing, the "ENO" output is reset.

Calls of code block are also shown in the network as boxes with EN/ENO mechanism.

See also

Rules for the use of LAD elements (Page 908)

Settings for LAD

Overview of the settings for LAD

Overview

The following table shows the settings that you can make:

Group	Setting	Description
Font	Size	Font size in program editor
View	Layout	Compact or wide Changes the vertical spacing between operands and other objects (such as operand and contact). The change becomes visible once the block is reopened.
	With absolute information	Additional display of the absolute addresses
Operand field	Maximum width	Maximum number of characters that can be entered horizontally in the operand field. This setting recalculates the layout of the networks.
	Maximum height	Maximum number of characters that can be entered vertically in the operand field. This setting recalculates the layout of the networks.

See also

Changing the settings (Page 901)

Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.

   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the settings for LAD (Page 901)

Working with networks

Using networks

Function

The user program is created in the block within networks. For a code block to be programmed, it must contain at least one network. To achieve a better overview of the user program, you can also subdivide your program into several networks.

See also

Entering the network title (Page 906)

Entering a network comment (Page 906)

Navigating networks (Page 907)

Inserting networks

Requirement

A block is open.

Procedure

To insert a new network, follow these steps:

1. Select the network after which you want to insert a new network.
2. Select the "Insert network" command in the shortcut menu.

Result

A new empty network is inserted into the block.

See also

Selecting networks (Page 903)

Copying and pasting networks (Page 904)

Deleting networks (Page 905)

Expanding and collapsing networks (Page 905)

Entering the network title (Page 906)

Entering a network comment (Page 906)

Navigating networks (Page 907)

Selecting networks

Requirements

A network is available.

Selecting a network

To select a network, follow these steps:

1. Click the title bar of the network that you want to select.

Selecting several networks

Proceed as follows to select several individual networks:

1. Press and hold down the <Ctrl> key.
2. Click all the networks that you want to select.

To select several successive networks, follow these steps:

1. Press and hold down the <Shift> key.
2. Click the first network that you want to select.
3. Click the last network that you want to select.

   The first and last networks and all those in between are selected.

See also

Inserting networks (Page 902)

Copying and pasting networks (Page 904)

Deleting networks (Page 905)

Expanding and collapsing networks (Page 905)

Entering the network title (Page 906)

Entering a network comment (Page 906)

Navigating networks (Page 907)

Copying and pasting networks

Copied networks can be pasted within the block or in another block. If the network is pasted into a block written in a different programming language, the programming language of the network is retained.

Requirements

A network is available.

Procedure

To copy and paste a network, follow these steps:

1. Select the network or networks to be copied.
2. Select "Copy" in the shortcut menu.
3. Select the network after which you want to paste in the copied network.
4. Select "Paste" in the shortcut menu.

See also

Inserting networks (Page 902)

Selecting networks (Page 903)

Deleting networks (Page 905)

Expanding and collapsing networks (Page 905)

Entering the network title (Page 906)

Entering a network comment (Page 906)

Navigating networks (Page 907)

Deleting networks

Requirement

A network is available.

Procedure

To delete a network, follow these steps:

1. Select the network that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Inserting networks (Page 902)

Selecting networks (Page 903)

Copying and pasting networks (Page 904)

Expanding and collapsing networks (Page 905)

Entering the network title (Page 906)

Entering a network comment (Page 906)

Navigating networks (Page 907)

Expanding and collapsing networks

Requirements

A network is available.

Opening and closing a network

To open a network, follow these steps:

1. Click on the right arrow in the network title bar.

To close a network, follow these steps:

1. Click on the down arrow in the network title bar.

Opening and closing all networks

To open and close all networks, follow these steps:

1. In the toolbar, click "Open all networks" or "Close all networks".

See also

Inserting networks (Page 902)

Selecting networks (Page 903)

Copying and pasting networks (Page 904)

Deleting networks (Page 905)

Entering the network title (Page 906)

Entering a network comment (Page 906)

Navigating networks (Page 907)

Entering the network title

The network title is the header of a network. The length of the network title is limited to one line.

Requirement

A network is available.

Procedure

To enter a network title, follow these steps:

1. Click on the title bar of the network.
2. Enter the network title.

See also

Using networks (Page 902)

Inserting networks (Page 902)

Selecting networks (Page 903)

Copying and pasting networks (Page 904)

Deleting networks (Page 905)

Expanding and collapsing networks (Page 905)

Entering a network comment (Page 906)

Navigating networks (Page 907)

Entering a network comment

You can use network comments to provide comments on the program contents of individual networks. For example, you can indicate the function of the network or draw attention to special characteristics.

Requirement

A network is available.

Procedure

To enter a network comment, follow these steps:

1. Click on the right arrow before the network title.
2. If the comment area is not visible, click "Network comments on/off" in the toolbar.

   The comment area is displayed.
3. Click "Comment" in the comment area.

   The "Comment" text passage is selected.
4. Enter the network comment.

See also

Using networks (Page 902)

Inserting networks (Page 902)

Selecting networks (Page 903)

Copying and pasting networks (Page 904)

Deleting networks (Page 905)

Expanding and collapsing networks (Page 905)

Entering the network title (Page 906)

Navigating networks (Page 907)

Navigating networks

You can navigate straight to a specific position within a block.

Procedure

To navigate to a specific position within a block, follow these steps:

1. Right-click on the white area of the programming window.
2. Select the "Go to > Network/line" command in the shortcut menu.

   The "Go to" dialog will open.
3. Enter the network to which you want to navigate.
4. Enter the line number of the network to which you want to navigate.
5. Confirm your entry with "OK".

Result

The relevant line will be displayed if this is possible. If the network or line requested does not exist, the last existing network or the last existing line in the network requested will be displayed.

See also

Using networks (Page 902)

Inserting networks (Page 902)

Selecting networks (Page 903)

Copying and pasting networks (Page 904)

Deleting networks (Page 905)

Expanding and collapsing networks (Page 905)

Entering the network title (Page 906)

Entering a network comment (Page 906)

Inserting LAD elements

Rules for the use of LAD elements

Rules

Note the following rules when inserting LAD elements:

- Every LAD network must terminate with a coil or a box. However, the following LAD elements must not be used to terminate a network:
  - Comparator boxes
  - Instructions for positive and negative RLO edge detection
- The starting point of the branch for a box connection must always be the power rail. Logic operations or other boxes can be present in the branch before the box.
- Only contacts can be inserted into simultaneous branches with preceding logic operations. The contact for negating the result of logic operation (-|NOT|-) is an exception here. The contact for negating the result of logic operation, as well as coils and boxes, can be used in simultaneous branches if they originate directly from the power rail.
- Constants (e.g. TRUE or FALSE) cannot be assigned to normally open or normally closed contacts. Instead, use the operands of the BOOL data type.
- Only one jump instruction can be inserted in each network.
- Only one jump label can be inserted in each network.
- Instructions with positive or negative edge detection may not be arranged directly at the left margin of the rung as they requires a prior logic operation.

Placement rules for S7-1200 CPUs

The following table sets out the instructions that can only be positioned at the end of the network:

Instruction		Prior logic operation required
Mnemonics	Name	
SET_BF	Set bit field	No
RESET_BF	Reset bit field	No
TP	Start pulse timer	Yes
TON	Start on-delay timer	Yes
TOF	Start off-delay timer	Yes
TONR	Time accumulator	Yes
JMP	Jump if RLO=1	No
JMPN	Jump if RLO=0	Yes
JMP_LIST	Define jump list	No
SWITCH	Jump distributor	No
RET	Return	No

Placement rules for S7-300/400 CPUs

The following table sets out the instructions that can only be positioned at the end of the network:

Instruction		Prior logic operation required
Mnemonics	Name	
S	Set output	Yes
R	Reset output	Yes
SP	Start pulse timer	Yes
SE	Start extended pulse timer	Yes
SD	Start on-delay timer	Yes
SS	Start retentive on-delay timer	Yes
SF	Start off-delay timer	Yes
SC	Set counter value	Yes
CU	Count up	Yes
CD	Count down	Yes
JMP	Jump if RLO=1	No
JMPN	Jump if RLO=0	Yes
RET	Return	No
OPN	Open global data block	No
OPNI	Open instance data block	No
CALL	Call block	No

Instruction		Prior logic operation required
Mnemonics	Name	
SAVE	Save RLO in BR bit	No
MCRA	Enable MCR range	No
MCRD	Disable MCR range	No
MCR<	Open MCR ranges	No
MCR>	Close MCR ranges	No

See also

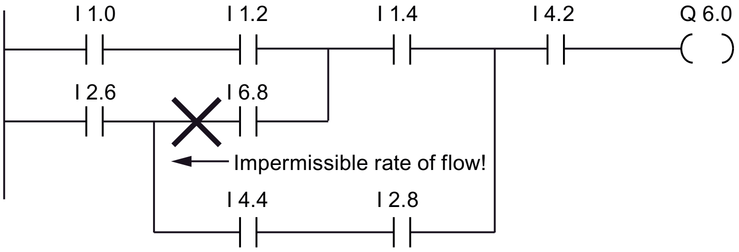
Prohibited interconnections in LAD (Page 910)

Overview of the LAD elements (Page 899)

Prohibited interconnections in LAD

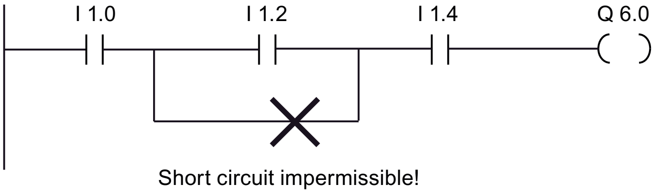
Power flow from right to left

No branches can be programmed that could result in a power flow in the reverse direction.



Short-circuit

No branches may be programmed that would cause a short-circuit.



Logic operations

The following rules apply to logic operations:

- Only Boolean inputs can be combined with preceding logic operations.
- Only the first Boolean output can be combined with a further logic operation.
- Only one complete logical path can exist per network. Paths that are not connected can be linked.

See also

Rules for the use of LAD elements (Page 908)

Inserting LAD elements using the "Instructions" task card

Requirement

A network is available.

Procedure

To insert a LAD element into a network using the "Instructions" task card, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to the LAD element that you want to insert.
3. Use drag-and-drop to move the element to the desired place in the network.

   If the element is an internal system function block (FB), the "Call option" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multi-instance", these are located in the block interface in the "Static" section.

Or:

1. Select the point in the network at which you want to insert the element.
2. Open the "Instructions" task card.
3. Double-click on the element you want to insert.

   If the element is an internal system function block (FB), the "Call option" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multi-instance", these are located in the block interface in the "Static" section.

Result

The selected LAD element is inserted with placeholders for the parameters.

Inserting LAD elements using an empty box

Requirement

A network is available.

Procedure

To insert an LAD element into a network using an empty box, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Empty box" in the "Basic instructions" palette.
3. Use a drag-and-drop operation to move the "Empty box" element to the desired place in the network.
4. Position the cursor over the triangle in the top right-hand corner of the empty box.

   A drop-down list is displayed.
5. Select the required LAD element from the drop-down list.

   If the element is an internal system function block (FB), the "Call option" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multi-instance", these are located in the block interface in the "Static" section.

Result

The empty box is changed to the respective LAD element. Placeholders are inserted for the parameters.

Selecting the data type of a LAD element

Selecting a data type

Introduction

Some instructions can be executed with several different data types. If one of these instructions is used in the program, a valid data type must be specified for the instruction at the specific point in the program. Data types for the inputs and outputs must be explicitly selected for some instructions.

Note

The valid data type (BOOL) for the tags on the enable input ENO and the enable output ENO is predefined by the system and cannot be changed.

The valid data types for an instruction are listed in the instruction drop-down list. You specify the data type of the instruction by selecting an entry from the drop-down list. If the data type of a specified tag differs from the data type of the instruction, the tag name will be displayed in red and a rollout with the corresponding error message will appear.

See also

Defining the data type of an instruction (Page 913)

Defining the data type of an instruction

Introduction

Some instructions can be executed with several different data types. If such instructions are to be inserted in the program, a data type for their execution must be defined at the specific point in the program in question.

Specifying the data type by means of the drop-down list

To define the data type of an instruction using the drop-down list, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.

   The instruction will be inserted at the selected point in the program. The entry "???" (undefined) is displayed in the drop-down list.

2. Click the yellow triangle in the upper corner of the drop-down list.

   The drop-down list will open to display the data types valid for the instruction.

3. Select a data type from the drop-down list.

   The selected data type is displayed.

4. If the instruction has two drop-down lists, select the data type for the instruction inputs in the left-hand drop-down list and the data type for the instruction outputs in the right-hand drop-down list.

Specifying data type by assigning tags

To define the data type of an instruction by assigning tags, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.

   The instruction will be inserted at the selected point in the program. The entry "???" (undefined) is displayed in the drop-down list.

2. At an input or output, specify a valid tag, the data type of which is to be applied as the instruction data type.

   The selected data type is displayed in the drop-down list.

3. Enter a valid tag at an input and a valid tag at an output if data types are to be defined for the instruction inputs and outputs. The tag specified at the input determines the data type of the inputs; the tag specified at the output determines the data type of the outputs of the instruction.

See also

Selecting a data type (Page 913)

Using favorites in LAD

Adding LAD elements to Favorites

Requirement

- A block is open.
- The multipane mode is set for the "Instructions" task card or the Favorites are also displayed in the editor.

Procedure

To add SCL instructions to the Favorites, follow these steps:

1. Open the "Instructions" task card.
2. Maximize the "Basic instructions" pane.
3. Navigate in the "Basic instructions" pane to the instruction that you want to add to the Favorites.
4. Drag-and-drop the instruction into the "Favorites" pane or into the Favorites area in the program editor.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Removing LAD elements from Favorites (Page 916)

Overview of the program editor (Page 863)

Inserting LAD elements using favorites

Requirement

- A block is open.
- Favorites are available.

Procedure

To insert an instruction into a program using Favorites, follow these steps:

1. Drag-and-drop the desired instruction from Favorites to the desired position.

Or:

1. Select the position in the program where you want to insert the instruction.
2. In the Favorites, click on the instruction you want to insert.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Removing LAD elements from Favorites (Page 916)

Overview of the program editor (Page 863)

Removing LAD elements from Favorites

Requirement

A code block is open.

Procedure

To remove instructions from Favorites, follow these steps:

1. Right-click on the instruction you want to remove.
2. Select the "Remove instruction" command in the shortcut menu.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Adding LAD elements to Favorites (Page 915)

Inserting LAD elements using favorites (Page 915)

Overview of the program editor (Page 863)

Insert block calls in LAD

Inserting block calls using a drag-and-drop operation

You can insert calls for existing functions (FC) and function blocks (FB) using a drag-and-drop operation from the project tree. If you call function blocks from other function blocks, you can either call them as single-instance or multi-instance blocks. If a function block is called as single instance, it will store its data in a data block of its own. If a function block is called as multi-instance, it will store its data in the instance data block of the calling function block.

Requirement

- A network is available.
- The block that is to be called is available.

Inserting a call of a function (FC)

To insert a call of a function (FC) into a network using a drag-and-drop operation, follow these steps:

1. Drag the function from the project tree to the required network.

Inserting a call for a function block (FB)

To insert a call for a function block (FB), follow these steps:

1. Drag the function block from the project tree to the required network.

   The "Call options" dialog opens.
2. Enter in the dialog whether you wish to call the block as single or multi-instance.
   – If you click on the "Single instance" button, you will have to enter a name in the "Name" text box for the data block that you want to assign to the function block.
   – If you click on the "Multi-instance" button, you will have to enter the name of the tag in the "Name in the interface" text box; this is the name that you use to enter the called function block as a static tag in the interface of the calling block.
3. Confirm your entries with "OK".

Result

The function or the function block is inserted with its parameters. You can then assign the parameters.

See also: Parameter transfer at block call (Page 707)

Note

If when calling a function block you specify an instance data block that does not exist, it will be created. If you have called a function block as a multi-instance, this will be entered as a static tag in the interface.

See also

Updating block calls in LAD (Page 918)

Correcting the call type (Page 919)

Single instances (Page 704)

Multi-instances (Page 706)

Updating block calls in LAD

If interface parameters of a called block are changed, the block call can no longer be executed correctly. You can avoid such inconsistent block calls by updating the block calls.

You have two options to update the block calls:

- Explicit updating in the program editor.

  The block calls in the open block will be updated.
- Implicit updating during compilation.

  All block calls in the program as well as the used PLC data types will be updated.

Update blocks in the program editor

To update a block call within a block, follow these steps:

1. Open the block in the program editor.
2. Click "Update inconsistent block calls" in the toolbar.

Or:

1. Open the block in the program editor.
2. Right-click on the instruction with the block call.
3. Select the "Update" command in the shortcut menu.

   The "Interface update" dialog opens. This dialog shows the differences between the block interface in use and the changed interface of the called block.
4. If you want to update the block call, click "OK". To cancel the update, click "Cancel".

Update block calls during compilation

Follow these steps to update all block calls and uses of PLC data types during compilation implicitly:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Inserting block calls using a drag-and-drop operation (Page 917)

Correcting the call type (Page 919)

Correcting the call type

Call type

There are two ways of calling function blocks:

- As a single instance
- As a multi-instance

See also: Call function blocks as single or multi-instances (Page 704)

You can modify a defined call type at any time.

Requirement

The user program contains a block call.

Procedure

To change the call type of a function block, follow these steps:

1. Open the code block and select the block call.
2. Select the "Change call type" command in the shortcut menu.

   The "Call options" dialog opens.
3. Click the "Single instance" or "Multi instance" button.
   – If you select the "Single instance" call type, enter a name for the data block that will be assigned to the function block.
   – If you select "Multi-instance" as the call type, enter the name of the tag in the "Name in the interface" text block with which the called function block will be entered as a static tag in the interface of the calling block.
4. Confirm your entries with "OK".

Note

The previous single and multiple instances will not be deleted automatically.

See also

Inserting block calls using a drag-and-drop operation (Page 917)

Updating block calls in LAD (Page 918)

Parameter transfer at block call (Page 707)

Inserting complex LAD instructions

Using the "Calculate" instruction

Requirement

A network is available.

Procedure

Proceed as follows to use the "Calculate" instruction:

1. Open the "Instructions" task card.
2. Navigate to "Math functions > CALCULATE" in the "Basic instructions" pane.
3. Use drag-and-drop to move the element to the desired place in the network.

   The instruction "Calculate" will be inserted for the data type with a placeholder expression and question mark.
4. Enter the data type for the calculation.

5. Enter the operands for the calculation.

   Note

   The calculation is run with the inputs of the "Calculate" instruction. If you want to use constants you must also insert appropriate inputs for them.

6. Click on the "Edit 'Calculate' instruction" button to replace the placeholder expression with the correct expression.

   The "Edit 'Calculate' instruction" dialog will open.

7. Enter the required expression in the "OUT:= " text box.

   Note

   In the "Example" area you can find an example of a valid expression and possible instructions that you can use.

   To determine a value with the help of Pythagoras' theorem, for example, enter "OUT := SQRT (SQR (IN1) + SQR (IN2))".

8. Confirm your entry with "OK".

See also

CALCULATE: Calculate (Page 1286)

Using free-form comments

Basic information on using free-form comments in LAD

Introduction

Free-form comments allow you to add comments to the source code for graphic programming languages similar to line comments for textual languages.

Free-form comments can be used for the following elements:

- Boxes
- Coils

See also

Inserting free-form comments (Page 922)

Editing free-form comments (Page 922)

Deleting free-form comments (Page 923)

Inserting free-form comments

Requirement

A network with instructions is available.

Procedure

To insert a free-form comment, proceed as follows:

1. Right-click on the instruction for which you want to insert a free-form comment.
2. Select the "Insert comment" command in the shortcut menu.

   A comment box with a standard comment opens. The comment box is connected by an arrow to the corresponding instruction.
3. Enter the required comment in the comment box.

See also

Basic information on using free-form comments in LAD (Page 921)

Editing free-form comments (Page 922)

Deleting free-form comments (Page 923)

Editing free-form comments

Introduction

Free-form comments can be edited as follows:

- Changing the comment text
- Changing the position and size of the comment box
- Attaching a comment to another element
- Showing and hiding free comments

Changing the comment text

To change the text of free-form comments, follow these steps:

1. Click on the comment box.
2. Enter the desired text.

Changing the position of the comment box

To change the positioning of the comment box, follow the steps below:

1. Left-click the comment box and keep the mouse button pressed.
2. Drag the comment box to the desired location.

Changing the size of the comment box

To change the size of the comment box, follow the steps below:

1. Click on the comment box.
2. Drag the comment box on the move handle in the lower right corner to the desired size.

Attaching a comment to another element

To attach a free-form comment to another element, follow these steps:

1. Left-click the point of the arrow that links the comment box with the instruction and keep the mouse button pressed.
2. Drag the arrow to the element to which you want to attach the comment. Possible insertion points are marked with a green square.
3. Release the mouse button.

Showing and hiding free comments

To show or hide a free-form comments, follow these steps:

1. Click the "Free-form comment on/off" button in the toolbar.

See also

Basic information on using free-form comments in LAD (Page 921)

Inserting free-form comments (Page 922)

Deleting free-form comments (Page 923)

Deleting free-form comments

Procedure

To delete a free-form comment, proceed as follows:

1. Right-click on the free-form comment that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Basic information on using free-form comments in LAD (Page 921)

Inserting free-form comments (Page 922)

Editing free-form comments (Page 922)

Editing LAD elements

Selecting LAD elements

You can select several individual elements or all elements in a network.

Requirement

LAD elements are available

Selecting several individual LAD elements

To select several individual LAD elements, follow these steps:

1. Press and hold down the <Ctrl> key.
2. Click on all the LAD elements you wish to select.
3. Now release the <Ctrl> key.

Selecting all LAD elements in a network

To select all LAD elements in a network, follow these steps:

1. Place the cursor in the network whose elements you wish to select.
2. Select the "Select all" in the "Edit" menu or press <Ctrl A>.

See also

Copying LAD elements (Page 925)

Cutting LAD elements (Page 925)

Pasting an LAD element from the clipboard (Page 926)

Replacing LAD elements (Page 926)

Inserting additional inputs and outputs in LAD elements (Page 927)

Removing inputs and outputs (Page 928)

Deleting LAD elements (Page 929)

Copying LAD elements

Requirement

An LAD element is available.

Procedure

To copy a LAD element, follow these steps:

1. Right-click the LAD element that you want to copy.
2. Select "Copy" in the shortcut menu.

Result

The LAD element will be copied and saved to the clipboard.

See also

Selecting LAD elements (Page 924)

Cutting LAD elements (Page 925)

Pasting an LAD element from the clipboard (Page 926)

Replacing LAD elements (Page 926)

Inserting additional inputs and outputs in LAD elements (Page 927)

Removing inputs and outputs (Page 928)

Deleting LAD elements (Page 929)

Cutting LAD elements

Requirement

An LAD element is available.

Cutting

To cut a LAD element, follow these steps:

1. Right-click the LAD element that you want to cut.
2. Select "Cut" in the shortcut menu.

Result

The LAD element will be cut and saved to the clipboard.

See also

Selecting LAD elements (Page 924)

Copying LAD elements (Page 925)

Pasting an LAD element from the clipboard (Page 926)

Replacing LAD elements (Page 926)

Inserting additional inputs and outputs in LAD elements (Page 927)

Removing inputs and outputs (Page 928)

Deleting LAD elements (Page 929)

Pasting an LAD element from the clipboard

Requirement

An LAD element is available.

Procedure

To paste an LAD element from the clipboard, follow these steps:

1. Copy a LAD element or cut a LAD element.
2. Right-click the point in the network where you want to paste the element.
3. Select "Paste" in the shortcut menu.

See also

Selecting LAD elements (Page 924)

Copying LAD elements (Page 925)

Cutting LAD elements (Page 925)

Replacing LAD elements (Page 926)

Inserting additional inputs and outputs in LAD elements (Page 927)

Removing inputs and outputs (Page 928)

Deleting LAD elements (Page 929)

Replacing LAD elements

You can easily exchange LAD elements with other LAD elements of the same type. This has the advantage that the parameters are retained and need not be entered again. For example, you can exchange normally open contacts and normally closed contacts or RS FlipFlop and SR FlipFlop.

Requirements

A network with at least one LAD element is present.

Procedure

To replace an LAD element with another LAD element, follow these steps:

1. Select the LAD element that you want to replace.
2. Position the cursor over the triangle in the top right-hand corner of the LAD element.

   A drop-down list is displayed.
3. From the drop-down list, select the LAD element that you want to use to replace the existing LAD element.

See also

Selecting LAD elements (Page 924)

Copying LAD elements (Page 925)

Cutting LAD elements (Page 925)

Pasting an LAD element from the clipboard (Page 926)

Inserting additional inputs and outputs in LAD elements (Page 927)

Removing inputs and outputs (Page 928)

Deleting LAD elements (Page 929)

Inserting additional inputs and outputs in LAD elements

Introduction

You can expand LAD elements which execute commutative arithmetic instructions by adding additional inputs. Such elements are, for example, the instructions "Add" (ADD) and "Multiply" (MUL). You can expand the MOVE and DEMUX instruction boxes by adding additional outputs.

Requirement

An LAD element is available that permits the insertion of additional inputs and outputs.

Inserting an additional input

To add an additional input to the box of a LAD element, follow these steps:

1. Right-click on an existing input of the LAD element.
2. Select "Insert input" in the shortcut menu.

   An additional input is added to the box of the LAD element.

Or:

1. Click on the yellow star symbol beside the last input in the instruction box.

   An additional input is added to the box of the LAD element.

Inserting an additional output

To add an additional output to the box of a LAD element, follow these steps:

1. Right-click on an existing output of the LAD element.
2. Select "Insert output" from the shortcut menu.

   An additional output is added to the box of the LAD element.

Or:

1. Click on the yellow star symbol beside the last input in the instruction box.

   An additional output is added to the box of the LAD element.

See also

Selecting LAD elements (Page 924)

Copying LAD elements (Page 925)

Cutting LAD elements (Page 925)

Pasting an LAD element from the clipboard (Page 926)

Replacing LAD elements (Page 926)

Removing inputs and outputs (Page 928)

Deleting LAD elements (Page 929)

Removing inputs and outputs

Introduction

Inputs and outputs which you have added to an instruction can be removed.

Requirement

An LAD element is available to which you have added additional inputs and outputs.

Remove input

To remove an input, follow these steps:

1. Select the input that you want to remove.
2. Select the "Delete" command in the shortcut menu.

   The input of the LAD element is removed.

Remove output

To remove an output, follow these steps:

1. Select the output that you want to remove.
2. Select the "Delete" command in the shortcut menu.

   The output of the LAD element will be removed.

See also

Selecting LAD elements (Page 924)

Copying LAD elements (Page 925)

Cutting LAD elements (Page 925)

Pasting an LAD element from the clipboard (Page 926)

Replacing LAD elements (Page 926)

Inserting additional inputs and outputs in LAD elements (Page 927)

Deleting LAD elements (Page 929)

Deleting LAD elements

Requirement

An LAD element is available.

Procedure

To delete a LAD element, follow these steps:

1. Right-click the LAD element that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Selecting LAD elements (Page 924)

Copying LAD elements (Page 925)

Cutting LAD elements (Page 925)

Pasting an LAD element from the clipboard (Page 926)

Replacing LAD elements (Page 926)

Inserting additional inputs and outputs in LAD elements (Page 927)

Removing inputs and outputs (Page 928)

Inserting operands into LAD instructions

Inserting operands

The character strings "<???>", "<??.?>" and "..." are inserted as placeholders for the parameters when an LAD element is inserted. The "<???>" and "<??.?>" strings displayed in red indicate parameters that need to be connected. The "..." string displayed in black indicates parameters that may be connected. "<??.?>" stands for Boolean placeholders.

Note

If you position the cursor over the placeholder, the expected data type will be displayed.

Requirement

An LAD element is available.

Procedure

To connect the parameters of a LAD element, follow these steps:

1. Double-click the placeholder of the parameter.

   An entry field opens, and the placeholder is selected.
2. Enter the appropriate parameter.

   Note

   If you enter the absolute address of a parameter that has already been defined, this absolute address will be changed to the symbolic name of the parameter as soon as the input is confirmed. If you have not yet defined the parameter, a new tag with this absolute address and the default name "Tag_<n>" will be entered in the PLC tag table. When you confirm your input, the absolute address will be replaced with the symbolic name "Tag_<n>".

3. Confirm the parameter with the Enter key.
4. If you have not yet defined the parameter, you can define it directly in the program editor using the shortcut menu.

   See also:

   Declaring PLC tags in the program editor (Page 832)

   Declaring local tags in the program editor (Page 882)

Or drag from it the PLC tag table:

1. In the project tree, select the "PLC tags" folder or open the PLC tag table.
2. If you have opened the PLC tag table, drag the symbol from the first column of the desired tag to the appropriate place in your program. If you have not opened the PLC tag table yet, open the detail view now. Drag the desired tag from the detail view to the appropriate place in your program.

Or drag from it the block interface:

1. Open the block interface.
2. Drag the required operand from the block interface to the instruction window.

Result

- If the syntax is error-free, the displayed parameter is black. The editor then jumps to the next placeholder.
- If there is an error in the syntax, the cursor stays in the entry field and a corresponding error message is displayed in the status line. If you press the Enter key again, the entry field is closed and the faulty entry is displayed in red italics.

Wiring hidden parameters

Introduction

Depending on the CPU used, you can use complex instructions in your program that are dispatched with the TIA portal. These instructions can contain parameters that are declared as hidden.

If an instruction contains hidden parameters, the instruction box has a small arrow on the lower edge. You can recognize hidden parameters by their white font.

You can show and wire the hidden parameters at any time.

Showing or hiding hidden parameters

To show or hide hidden parameters, follow these steps:

1. Click on the down arrow at the bottom edge of the instruction box to show hidden parameters.
2. Click on the up arrow at the bottom edge of the instruction box to hide hidden parameters.

Wiring hidden parameters

To wire parameters, follow these steps:

1. Wire the hidden parameters like normally visible parameters.

   The hidden parameter is transformed into a visible parameter.

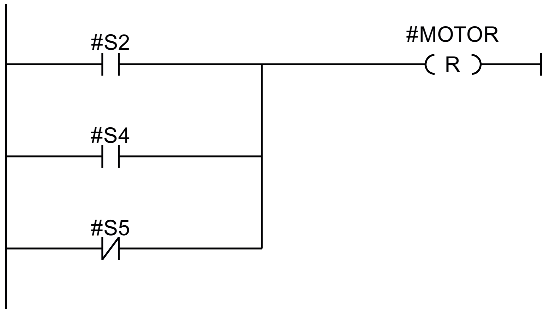
Branches in LAD

Basic information on branches in LAD

Definition

You use branches to program parallel circuits with the Ladder Logic (LAD) programming language. Branches are inserted in the main rung. You can insert several contacts into the branch and thus achieve a parallel circuit of series connections. This allows you to program complex ladder logic.

The figure below shows an example of the use of branches:



MOTOR carries signal 1, if one of the following conditions is fulfilled:

- Signal 1 is pending on S2 or S4
- Signal 0 is pending on S5.

See also

Rules for branches in LAD (Page 933)

Inserting branches into the LAD network (Page 933)

Closing branches in the LAD network (Page 934)

Deleting branches in LAD networks (Page 935)

Rules for branches in LAD

Rules

The following rules apply to simultaneous branches:

- A simultaneous branch can only be inserted if the main branch already contains an LAD element.
- Simultaneous branches are opened downwards or are connected directly to the power rail. They are terminated upwards.
- Simultaneous branches are opened after the selected LAD element.
- Simultaneous branches are terminated after the selected LAD element.
- To delete a simultaneous branch, you must delete all LAD elements of this branch. When the last LAD element is removed from the branch, the rest of the branch is also removed.

See also

Basic information on branches in LAD (Page 932)

Inserting branches into the LAD network (Page 933)

Deleting branches in LAD networks (Page 935)

Closing branches in the LAD network (Page 934)

Inserting branches into the LAD network

You can create several branches in a network.

Requirement

- A network is available.
- The network contains elements.

Procedure

To insert a new branch in a network, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Open branches" in the "Simple instructions" palette.
3. Use drag-and-drop to move the element to the desired place in the network.

   If you want to connect the new branch directly to the power rail, drag the element to the power rail.

See also

Basic information on branches in LAD (Page 932)

Rules for branches in LAD (Page 933)

Deleting branches in LAD networks (Page 935)

Closing branches in the LAD network

Branches must be closed again at suitable places. If necessary, branches will be arranged so that they do not cross each other.

Requirement

A branch is available.

Procedure

To close an open branch, follow these steps:

1. Select the open branch.
2. Press and hold down the left mouse button.

   A dashed line will appear as soon as the cursor is moved.
3. Drag the dashed line to a suitable place on the network. Permissible connections are indicated by green lines.
4. Release the left mouse button.

See also

Basic information on branches in LAD (Page 932)

Rules for branches in LAD (Page 933)

Deleting branches in LAD networks

Requirement

A branch is available.

Procedure

To delete a branch, follow these steps:

1. Select the connection line that links the branch to the main branch.
2. Select the "Delete" command in the shortcut menu.

See also

Basic information on branches in LAD (Page 932)

Rules for branches in LAD (Page 933)

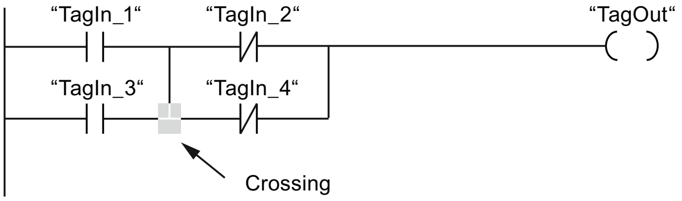
Inserting branches into the LAD network (Page 933)

Crossings in LAD

Basic information on crossings in LAD

Definition

A crossing is a place in a LAD network where one branch is closed and at the same time another branch is opened.



"TagOut" receives signal 1, if the following two conditions are met:

- "TagIn_1" or "TagIn_3" has signal 1
- "TagIn_2" or "TagIn_4" has signal 0

Inserting crossings

You can insert crossings in a LAD network by creating connections between the main branch and an additional branch or between different branches.

Requirements

A branch is available.

Procedure

To insert a new crossing in an LAD network, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Open branches" in the "Simple instructions" palette.
3. Drag the element behind the existing branch.
4. Insert any element into the open branch.
5. Click the arrow of the open branch after the inserted element.
6. Hold down the left mouse button and drag the dashed connecting line to the main branch.
7. Release the left mouse button.

See also

Rearranging crossings (Page 936)

Deleting crossings (Page 937)

Inserting branches into the LAD network (Page 933)

Rearranging crossings

Requirement

A crossing is available.

Procedure

To rearrange a connection, follow these steps:

1. Select the connection line that defines the crossings in the respective branches.
2. Select the "Delete" command in the shortcut menu.
3. Open the "Instructions" task card.
4. Navigate to "General > Open branches" in the "Simple instructions" palette.
5. Use a drag-and-drop operation to move the element to the place in the network where you want to insert the new crossing.
6. Click on the arrow for the open branch.
7. Hold down the left mouse button and drag the dashed connecting line to the subsidiary branch in which you wish to insert the new crossing.
8. Release the left mouse button.

See also

Inserting crossings (Page 935)

Deleting crossings (Page 937)

Deleting crossings

Requirement

A crossing is available.

Procedure

To delete a crossing, follow these steps:

1. Select the connection line that defines the crossings in the respective branches.
2. Select the "Delete" command in the shortcut menu.

See also

Inserting crossings (Page 935)

Rearranging crossings (Page 936)

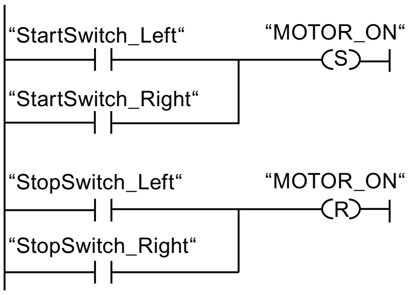
Rungs in LAD

Basic information on rungs in LAD

Using rungs

The program is mapped in one or more networks. A network contains a power rail on the left where one or more rungs originate. The binary signal scans are arranged in the form of contacts on the rungs. The serial arrangement of the elements on a rung creates a series connection; arrangement on simultaneous branches creates a parallel connection. A rung is closed by a coil or a box in which the result of logic operation will be written.

The figure below shows an example of the use of several rungs within a network:



Rules

Remember the following rules when using several rungs:

- Connections are not permitted between rungs.
- Only one jump instruction is permissible per network. The positioning rules for jump instructions remain valid.

Running rungs

Rungs and networks are executed from top to bottom and from left to right. This means that the first instruction in the first rung of the first network is processed first. All instructions of this rung are then processed. After this come all other rungs of the first network. The next network is processed only after all rungs have first been run.

Differences between branches and rungs

The difference between branches and rungs is that the rungs are independent branches that can also stand in a different network. Branches, on the other hand, permit the programming of a parallel connection.

See also

Insert rung (Page 938)

Deleting a rung (Page 939)

Insert rung

Requirement

- A block is open.
- A network is available.

Procedure

To insert a new rung in a network, proceed as follows:

1. Insert any coil on the power rail.

   A new rung will be inserted and the coil positioned at the end of the rung.
2. Insert additional instructions in the new rung.

See also

Basic information on rungs in LAD (Page 937)

Deleting a rung (Page 939)

Deleting a rung

Requirement

A rung is available.

Procedure

To delete a rung, proceed as follows:

1. Hold down the left mouse button and draw a frame around the rung. At the same time, make sure that you select all instructions. Alternatively, you can hold down the <Shift> key and select the first the last instruction of the rung.
2. Right-click on one of the instructions in the rung.
3. Select the "Delete" command in the shortcut menu.

See also

Basic information on rungs in LAD (Page 937)

Insert rung (Page 938)

Creating FBD programs

Basic information on FBD

FBD programming language

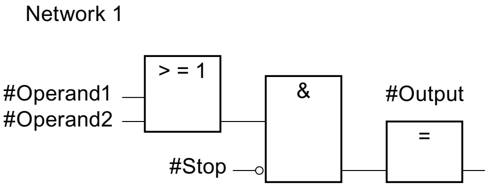
Overview of the Function Block Diagram (FBD) programming language

FBD is a graphical programming language. The representation is based on electronic circuit systems.

The program is mapped in one or more networks. A network contains one or more logic operation paths. The binary signal scans are linked by boxes. The representation of the logic is based on the graphical logic symbols used in Boolean algebra.

Example of networks in FBD

The following figure shows an FBD network with AND and OR boxes and an assignment:



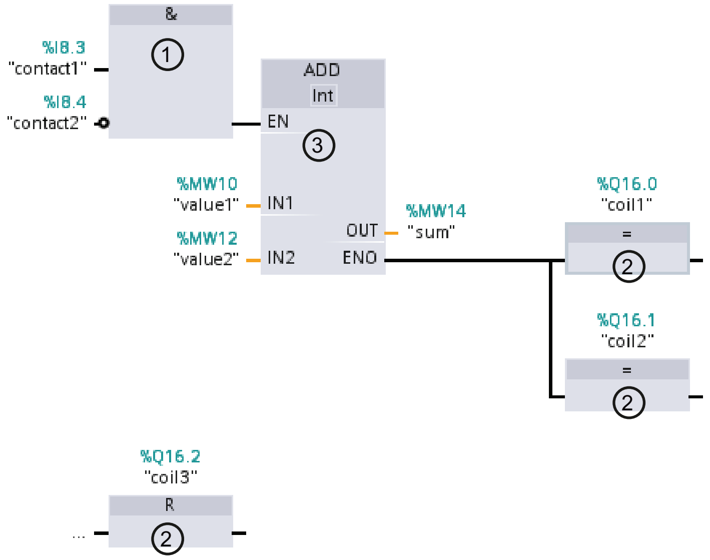
Overview of the FBD elements

FBD elements

An FBD program consists of separate elements that are linked by means of a binary signal flow. Most program elements must be supplied with tags.

A FBD network is programmed from left to right.

For example, the following figure shows elements of an FBD network:



1) Binary function

2) Standard box

3) Complex box

Binary functions

You can use binary functions to query binary operands and to combine their signal states. The following operations are examples of binary functions: "AND operation", "OR operation" and "EXCLUSIVE OR operation".

Standard boxes:

You can use standard boxes to control binary operands, perform RLO edge detection or execute jump functions in the program. Standard boxes generally have only one single input.

Complex boxes

Complex boxes represent program elements with complex functions. The empty box is an exception. You can use the empty box as a placeholder in which you can select the required instruction.

The following types of boxes are available to you in an FBD program:

- Complex boxes without EN/ENO mechanism:
  A box is executed independent of the signal state at the box inputs. The error status of the processing cannot be queried.
- Boxes with EN/ENO mechanism:
  A box is only executed if the enabling input "EN" carries the signal state "1". If the box is processed correctly, the "ENO" enable output has signal state "1". If an error occurs during processing, the "ENO" output is reset.
  If the EN enable input is not interconnected, the box is always executed.

Calls of code block are also shown in the network as complex boxes with EN/ENO mechanism.

Settings for FBD

Overview of the settings for FBD

Overview

The following table shows the settings that you can make:

Group	Setting	Description
Font	Size	Font size in program editor
View	Layout	Compact or wide Changes the vertical spacing between operands and other objects (such as operand and contact). The change becomes visible once the block is reopened.
	With absolute information	Additional display of the absolute addresses
Operand field	Maximum width	Maximum number of characters that can be entered horizontally in the operand field. This setting recalculates the layout of the networks.
	Maximum height	Maximum number of characters that can be entered vertically in the operand field. This setting recalculates the layout of the networks.

See also

Changing the settings (Page 942)

Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.

   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the settings for FBD (Page 941)

Working with networks

Using networks

Function

The user program is created in the block within networks. For a code block to be programmed, it must contain at least one network. To achieve a better overview of the user program, you can also subdivide your program into several networks.

See also

Entering the network title (Page 946)

Entering a network comment (Page 946)

Navigating networks (Page 947)

Inserting networks

Requirement

A block is open.

Procedure

To insert a new network, follow these steps:

1. Select the network after which you want to insert a new network.
2. Select the "Insert network" command in the shortcut menu.

Result

A new empty network is inserted into the block.

See also

Entering the network title (Page 946)

Entering a network comment (Page 946)

Navigating networks (Page 947)

Selecting networks

Requirements

A network is available.

Selecting a network

To select a network, follow these steps:

1. Click the title bar of the network that you want to select.

Selecting several networks

Proceed as follows to select several individual networks:

1. Press and hold down the <Ctrl> key.
2. Click all the networks that you want to select.

To select several successive networks, follow these steps:

1. Press and hold down the <Shift> key.
2. Click the first network that you want to select.
3. Click the last network that you want to select.

   The first and last networks and all those in between are selected.

See also

Inserting networks (Page 942)

Entering the network title (Page 946)

Entering a network comment (Page 946)

Navigating networks (Page 947)

Copying and pasting networks

Copied networks can be pasted within the block or in another block. If the network is pasted into a block written in a different programming language, the programming language of the network is retained.

Requirements

A network is available.

Procedure

To copy and paste a network, follow these steps:

1. Select the network or networks to be copied.
2. Select "Copy" in the shortcut menu.
3. Select the network after which you want to paste in the copied network.
4. Select "Paste" in the shortcut menu.

See also

Inserting networks (Page 942)

Selecting networks (Page 943)

Entering the network title (Page 946)

Entering a network comment (Page 946)

Navigating networks (Page 947)

Deleting networks

Requirement

A network is available.

Procedure

To delete a network, follow these steps:

1. Select the network that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Inserting networks (Page 942)

Selecting networks (Page 943)

Copying and pasting networks (Page 944)

Entering the network title (Page 946)

Entering a network comment (Page 946)

Navigating networks (Page 947)

Expanding and collapsing networks

Requirements

A network is available.

Opening and closing a network

To open a network, follow these steps:

1. Click on the right arrow in the network title bar.

To close a network, follow these steps:

1. Click on the down arrow in the network title bar.

Opening and closing all networks

To open and close all networks, follow these steps:

1. In the toolbar, click "Open all networks" or "Close all networks".

See also

Inserting networks (Page 942)

Selecting networks (Page 943)

Copying and pasting networks (Page 944)

Deleting networks (Page 945)

Entering the network title (Page 946)

Entering a network comment (Page 946)

Navigating networks (Page 947)

Entering the network title

The network title is the header of a network. The length of the network title is limited to one line.

Requirement

A network is available.

Procedure

To enter a network title, follow these steps:

1. Click on the title bar of the network.
2. Enter the network title.

See also

Using networks (Page 942)

Inserting networks (Page 942)

Selecting networks (Page 943)

Copying and pasting networks (Page 944)

Deleting networks (Page 945)

Expanding and collapsing networks (Page 945)

Entering a network comment (Page 946)

Navigating networks (Page 947)

Entering a network comment

You can use network comments to provide comments on the program contents of individual networks. For example, you can indicate the function of the network or draw attention to special characteristics.

Requirement

A network is available.

Procedure

To enter a network comment, follow these steps:

1. Click on the right arrow before the network title.
2. If the comment area is not visible, click "Network comments on/off" in the toolbar.

   The comment area is displayed.
3. Click "Comment" in the comment area.

   The "Comment" text passage is selected.
4. Enter the network comment.

See also

Using networks (Page 942)

Inserting networks (Page 942)

Selecting networks (Page 943)

Copying and pasting networks (Page 944)

Deleting networks (Page 945)

Expanding and collapsing networks (Page 945)

Entering the network title (Page 946)

Navigating networks (Page 947)

Navigating networks

You can navigate straight to a specific position within a block.

Procedure

To navigate to a specific position within a block, follow these steps:

1. Right-click on the white area of the programming window.
2. Select the "Go to > Network/line" command in the shortcut menu.

   The "Go to" dialog will open.
3. Enter the network to which you want to navigate.
4. Enter the line number of the network to which you want to navigate.
5. Confirm your entry with "OK".

Result

The relevant line will be displayed if this is possible. If the network or line requested does not exist, the last existing network or the last existing line in the network requested will be displayed.

See also

Using networks (Page 942)

Inserting networks (Page 942)

Selecting networks (Page 943)

Copying and pasting networks (Page 944)

Deleting networks (Page 945)

Expanding and collapsing networks (Page 945)

Entering the network title (Page 946)

Entering a network comment (Page 946)

Inserting FBD elements

Rules for the use of FBD elements

Rules

Note the following rules when inserting FBD elements:

- An FBD network can consist of several elements. All elements of a logic path must be linked to each other according to IEC 61131-3.
- Standard boxes (flip flops, counters, timers, math operations, etc.) can be added as output to boxes with binary logic operations (for example AND, OR). Comparison boxes are excluded from this rule.
- Only Boolean inputs in an instruction can be combined with preceding logic operations.
- Only the bottom Boolean output in an instruction can be combined with an additional logic operation.
- Enable input "EN" and enable output "ENO" can be connected to boxes, but this is not mandatory.
- Constants (e.g. TRUE or FALSE) cannot be assigned to binary logic operations. Instead of this, use tags of the BOOL data type.
- Only one jump instruction can be inserted in each network.
- Only one jump label can be inserted in each network.
- Instructions for positive or negative RLO edge detection may not be arranged right at the left of the network as this requires a prior logic operation.

Placement rules for S7-1200 CPUs

The following table sets out the instructions that can only be positioned at the end of the network:

Instruction		Prior logic operation required
Mnemonics	Name	
SET_BF	Set bit field	No
RESET_BF	Reset bit field	No
TP	Start pulse timer	Yes
TON	Start on-delay timer	Yes
TOF	Start off-delay timer	Yes
TONR	Time accumulator	Yes
JMP	Jump if RLO=1	No
JMPN	Jump if RLO=0	Yes
JMP_LIST	Define jump list	No
SWITCH	Jump distributor	No
RET	Return	No

Placement rules for S7-300/400 CPUs

The following table sets out the instructions that can only be positioned at the end of the network:

Instruction		Prior logic operation required
Mnemonics	Name	
S	Set output	Yes
R	Reset output	Yes
SP	Start pulse timer	Yes
SE	Start extended pulse timer	Yes
SD	Start on-delay timer	Yes
SS	Start retentive on-delay timer	Yes
SF	Start off-delay timer	Yes
SC	Set counter value	Yes
CU	Count up	Yes
CD	Count down	Yes
JMP	Jump if RLO=1	No
JMPN	Jump if RLO=0	Yes
RET	Return	No
OPN	Open global data block	No
OPNI	Open instance data block	No
CALL	Call block	No

Instruction		Prior logic operation required
Mnemonics	Name	
SAVE	Save RLO in BR bit	No
MCRA	Enable MCR range	No
MCRD	Disable MCR range	No
MCR<	Open MCR ranges	No
MCR>	Close MCR ranges	No

Inserting FBD elements using the "Instructions" task card

Requirement

A network is available.

Procedure

To insert FBD elements into a network using the "Instructions" task card, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to the FBD element that you want to insert.
3. Use drag-and-drop to move the element to the desired place in the network.

   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multi-instance", these are located in the block interface in the "Static" section.

Or:

1. Select the point in the network at which you want to insert the element.
2. Open the "Instructions" task card.
3. Double-click on the element you want to insert.

   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multi-instance", these are located in the block interface in the "Static" section.

Result

The selected FBD element is inserted with dummy entries for the parameters.

See also

Rules for the use of FBD elements (Page 948)

Inserting FBD elements using an empty box

Requirement

A network is available.

Procedure

To insert FBD elements into a network using an empty box, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Empty box" in the "Basic instructions" palette.
3. Use a drag-and-drop operation to move the "Empty box" element to the desired place in the network.
4. Position the cursor over the triangle in the top right-hand corner of the empty box.

   A drop-down list is displayed.
5. Select the desired FBD element from the drop-down list.

   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multi-instance", these are located in the block interface in the "Static" section.

Result

The empty box is changed to the respective FBD element. Placeholders are inserted for the parameters.

Selecting the data type of an FBD element

Selecting a data type

Introduction

Some instructions can be executed with several different data types. If one of these instructions is used in the program, a valid data type must be specified for the instruction at the specific point in the program. Data types for the inputs and outputs must be explicitly selected for some instructions.

Note

The valid data type (BOOL) for the tags on the enable input ENO and the enable output ENO is predefined by the system and cannot be changed.

The valid data types for an instruction are listed in the instruction drop-down list. You specify the data type of the instruction by selecting an entry from the drop-down list. If the data type of a specified tag differs from the data type of the instruction, the tag name will be displayed in red and a rollout with the corresponding error message will appear.

Defining the data type of an instruction

Introduction

Some instructions can be executed with several different data types. If such instructions are to be inserted in the program, a data type for their execution must be defined at the specific point in the program in question.

Specifying the data type by means of the drop-down list

To define the data type of an instruction using the drop-down list, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.

   The instruction will be inserted at the selected point in the program. The entry "???" (undefined) is displayed in the drop-down list.

2. Click the yellow triangle in the upper corner of the drop-down list.

   The drop-down list will open to display the data types valid for the instruction.

3. Select a data type from the drop-down list.

   The selected data type is displayed.

4. If the instruction has two drop-down lists, select the data type for the instruction inputs in the left-hand drop-down list and the data type for the instruction outputs in the right-hand drop-down list.

Specifying data type by assigning tags

To define the data type of an instruction by assigning tags, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.

   The instruction will be inserted at the selected point in the program. The entry "???" (undefined) is displayed in the drop-down list.
2. At an input or output, specify a valid tag, the data type of which is to be applied as the instruction data type.

   The selected data type is displayed in the drop-down list.
3. Enter a valid tag at an input and a valid tag at an output if data types are to be defined for the instruction inputs and outputs. The tag specified at the input determines the data type of the inputs; the tag specified at the output determines the data type of the outputs of the instruction.

Using favorites

Adding FBD elements to Favorites

Requirement

- A block is open.
- The multipane mode is set for the "Instructions" task card or the Favorites are also displayed in the editor.

Procedure

To add SCL instructions to the Favorites, follow these steps:

1. Open the "Instructions" task card.
2. Maximize the "Basic instructions" pane.
3. Navigate in the "Basic instructions" pane to the instruction that you want to add to the Favorites.
4. Drag-and-drop the instruction into the "Favorites" pane or into the Favorites area in the program editor.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Removing FBD elements from Favorites (Page 954)

Overview of the program editor (Page 863)

Inserting FBD elements using favorites

Requirement

- A block is open.
- Favorites are available.

Procedure

To insert an instruction into a program using Favorites, follow these steps:

1. Drag-and-drop the desired instruction from Favorites to the desired position.

Or:

1. Select the position in the program where you want to insert the instruction.
2. In the Favorites, click on the instruction you want to insert.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Overview of the program editor (Page 863)

Removing FBD elements from Favorites (Page 954)

Removing FBD elements from Favorites

Requirement

A code block is open.

Procedure

To remove instructions from Favorites, follow these steps:

1. Right-click on the instruction you want to remove.
2. Select the "Remove instruction" command in the shortcut menu.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Adding FBD elements to Favorites (Page 953)

Inserting FBD elements using favorites (Page 954)

Overview of the program editor (Page 863)

Inserting block calls in FBD

Inserting block calls using a drag-and-drop operation

You can insert calls for existing functions (FC) and function blocks (FB) using a drag-and-drop operation from the project tree. If you call function blocks from other function blocks, you can either call them as single-instance or multi-instance blocks. If a function block is called as single instance, it will store its data in a data block of its own. If a function block is called as multi-instance, it will store its data in the instance data block of the calling function block.

Requirement

- A network is available.
- The block that is to be called is available.

Inserting a call of a function (FC)

To insert a call of a function (FC) into a network using a drag-and-drop operation, follow these steps:

1. Drag the function from the project tree to the required network.

Inserting a call for a function block (FB)

To insert a call for a function block (FB), follow these steps:

1. Drag the function block from the project tree to the required network.

   The "Call options" dialog opens.

2. Enter in the dialog whether you wish to call the block as single or multi-instance.

   – If you click on the "Single instance" button, you will have to enter a name in the "Name" text box for the data block that you want to assign to the function block.

   – If you click on the "Multi-instance" button, you will have to enter the name of the tag in the "Name in the interface" text box; this is the name that you use to enter the called function block as a static tag in the interface of the calling block.

3. Confirm your entries with "OK".

Result

The function or the function block is inserted with its parameters. You can then assign the parameters.

See also: Parameter transfer at block call (Page 707)

Note

If when calling a function block you specify an instance data block that does not exist, it will be created. If you have called a function block as a multi-instance, this will be entered as a static tag in the interface.

See also

Updating block calls in FBD (Page 956)

Correcting the call type (Page 957)

Single instances (Page 704)

Multi-instances (Page 706)

Updating block calls in FBD

If interface parameters of a called block are changed, the block call can no longer be executed correctly. You can avoid such inconsistent block calls by updating the block calls.

You have two options to update the block calls:

- Explicit updating in the program editor.

  The block calls in the open block will be updated.

- Implicit updating during compilation.

  All block calls in the program as well as the used PLC data types will be updated.

Update blocks in the program editor

To update a block call within a block, follow these steps:

1. Open the block in the program editor.
2. Click "Update inconsistent block calls" in the toolbar.

Or:

1. Open the block in the program editor.
2. Right-click on the instruction with the block call.
3. Select the "Update" command in the shortcut menu.

   The "Interface update" dialog opens. This dialog shows the differences between the block interface in use and the changed interface of the called block.
4. If you want to update the block call, click "OK". To cancel the update, click "Cancel".

Update block calls during compilation

Follow these steps to update all block calls and uses of PLC data types during compilation implicitly:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Inserting block calls using a drag-and-drop operation (Page 955)

Correcting the call type (Page 957)

Correcting the call type

Call type

There are two ways of calling function blocks:

- As a single instance
- As a multi-instance

See also: Call function blocks as single or multi-instances (Page 704)

You can modify a defined call type at any time.

Requirement

The user program contains a block call.

Procedure

To change the call type of a function block, follow these steps:

1. Open the code block and select the block call.
2. Select the "Change call type" command in the shortcut menu.

   The "Call options" dialog opens.
3. Click the "Single instance" or "Multi instance" button.
   - If you select the "Single instance" call type, enter a name for the data block that will be assigned to the function block.
   - If you select "Multi-instance" as the call type, enter the name of the tag in the "Name in the interface" text block with which the called function block will be entered as a static tag in the interface of the calling block.
4. Confirm your entries with "OK".

Note

The previous single and multiple instances will not be deleted automatically.

See also

Inserting block calls using a drag-and-drop operation (Page 955)

Updating block calls in FBD (Page 956)

Inserting complex FBD instructions

Using the "Calculate" instruction

Requirement

A network is available.

Procedure

Proceed as follows to use the "Calculate" instruction:

1. Open the "Instructions" task card.
2. Navigate to "Math functions > CALCULATE" in the "Basic instructions" pane.
3. Use drag-and-drop to move the element to the desired place in the network.

   The instruction "Calculate" will be inserted for the data type with a placeholder expression and question mark.
4. Enter the data type for the calculation.

5. Enter the operands for the calculation.

   Note

   The calculation is run with the inputs of the "Calculate" instruction. If you want to use constants you must also insert appropriate inputs for them.

   ---

6. Click on the "Edit 'Calculate' instruction" button to replace the placeholder expression with the correct expression.

   The "Edit 'Calculate' instruction" dialog will open.

7. Enter the required expression in the "OUT:= " text box.

   Note

   In the "Example" area you can find an example of a valid expression and possible instructions that you can use.

   To determine a value with the help of Pythagoras' theorem, for example, enter "OUT := SQRT (SQR (IN1) + SQR (IN2))".

   ---

8. Confirm your entry with "OK".

See also

CALCULATE: Calculate (Page 1455)

Using free-form comments

Basic information on using free comments in FBD

Introduction

Free-form comments allow you to add comments to the source code for graphic programming languages similar to line comments for textual languages.

Free-form comments can be used for all non-binary boxes.

See also

Inserting free-form comments (Page 960)

Editing free-form comments (Page 960)

Deleting free-form comments (Page 961)

Inserting free-form comments

Requirement

A network with instructions is available.

Procedure

To insert a free-form comment, proceed as follows:

1. Right-click on the instruction for which you want to insert a free-form comment.
2. Select the "Insert comment" command in the shortcut menu.

   A comment box with a standard comment opens. The comment box is connected by an arrow to the corresponding instruction.
3. Enter the required comment in the comment box.

See also

Basic information on using free comments in FBD (Page 959)

Editing free-form comments (Page 960)

Deleting free-form comments (Page 961)

Editing free-form comments

Introduction

Free-form comments can be edited as follows:

- Changing the comment text
- Changing the position and size of the comment box
- Attaching a comment to another element
- Showing and hiding free comments

Changing the comment text

To change the text of free-form comments, follow these steps:

1. Click on the comment box.
2. Enter the desired text.

Changing the position of the comment box

To change the positioning of the comment box, follow the steps below:

1. Left-click the comment box and keep the mouse button pressed.
2. Drag the comment box to the desired location.

Changing the size of the comment box

To change the size of the comment box, follow the steps below:

1. Click on the comment box.
2. Drag the comment box on the move handle in the lower right corner to the desired size.

Attaching a comment to another element

To attach a free-form comment to another element, follow these steps:

1. Left-click the point of the arrow that links the comment box with the instruction and keep the mouse button pressed.
2. Drag the arrow to the element to which you want to attach the comment. Possible insertion points are marked with a green square.
3. Release the mouse button.

Showing and hiding free comments

To show or hide a free-form comments, follow these steps:

1. Click the "Free-form comment on/off" button in the toolbar.

See also

Basic information on using free comments in FBD (Page 959)

Inserting free-form comments (Page 960)

Deleting free-form comments (Page 961)

Deleting free-form comments

Procedure

To delete a free-form comment, proceed as follows:

1. Right-click on the free-form comment that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Basic information on using free comments in FBD (Page 959)

Inserting free-form comments (Page 960)

Editing free-form comments (Page 960)

Editing FBD elements

Selecting FBD elements

You can select several individual elements or all elements in a network.

Requirement

FBD elements are available

Selecting several individual FBD elements

To select several individual FBD elements, follow these steps:

1. Press and hold down the <Ctrl> key.
2. Click on all the FBD elements you wish to select.
3. Now release the <Ctrl> key.

Selecting all FBD elements in a network

To select all FBD elements in a network, follow these steps:

1. Place the cursor in the network whose elements you wish to select.
2. Select the "Select all" in the "Edit" menu or press <Ctrl A>.

See also

Copying FBD elements (Page 963)

Cutting FBD elements (Page 963)

Pasting an FBD element from the clipboard (Page 964)

Replacing FBD elements (Page 964)

Adding additional inputs and outputs to FBD elements (Page 965)

Removing instruction inputs and outputs (Page 966)

Deleting FBD elements (Page 967)

Copying FBD elements

Requirement

An FBD element is available.

Procedure

To copy an FBD element, follow these steps:

1. Right-click the FBD element that you want to copy.
2. Select "Copy" in the shortcut menu.

Result

The FBD element will be copied and saved to the clipboard.

See also

Selecting FBD elements (Page 962)

Cutting FBD elements (Page 963)

Pasting an FBD element from the clipboard (Page 964)

Replacing FBD elements (Page 964)

Adding additional inputs and outputs to FBD elements (Page 965)

Removing instruction inputs and outputs (Page 966)

Deleting FBD elements (Page 967)

Cutting FBD elements

Requirement

An FBD element is available.

Cutting

To cut an FBD element, follow these steps:

1. Right-click the FBD element that you want to cut.
2. Select "Cut" in the shortcut menu.

Result

The FBD element will be cut and saved to the clipboard.

See also

Selecting FBD elements (Page 962)

Copying FBD elements (Page 963)

Pasting an FBD element from the clipboard (Page 964)

Replacing FBD elements (Page 964)

Adding additional inputs and outputs to FBD elements (Page 965)

Removing instruction inputs and outputs (Page 966)

Deleting FBD elements (Page 967)

Pasting an FBD element from the clipboard

Requirement

An FBD element is available.

Procedure

To paste an FBD element from the clipboard, follow these steps:

1. Copy an FBD element or cut an FBD element.
2. Right-click the point in the network where you want to paste the element.
3. Select "Paste" in the shortcut menu.

See also

Selecting FBD elements (Page 962)

Copying FBD elements (Page 963)

Cutting FBD elements (Page 963)

Replacing FBD elements (Page 964)

Adding additional inputs and outputs to FBD elements (Page 965)

Removing instruction inputs and outputs (Page 966)

Deleting FBD elements (Page 967)

Replacing FBD elements

You can easily exchange FBD elements with other FBD elements of the same type. This has the advantage that the parameters are retained and need not be entered again. For example, you can exchange OR and AND, RS-FlipFlop and SR-FlipFlop, comparison functions or jump instructions.

Requirements

A network with at least one FBD element is present.

Procedure

To replace an FBD element with another FBD element, follow these steps:

1. Select the FBD element that you want to replace.

   If elements compatible with the selected FBD element are available, a triangle will appear in the upper right-hand corner of the element.
2. Position the cursor above the triangle of the FBD element.

   A drop-down list is displayed.
3. From the drop-down list, select the FBD element that you want to use to replace the existing FBD element.

See also

Selecting FBD elements (Page 962)

Copying FBD elements (Page 963)

Cutting FBD elements (Page 963)

Pasting an FBD element from the clipboard (Page 964)

Adding additional inputs and outputs to FBD elements (Page 965)

Removing instruction inputs and outputs (Page 966)

Deleting FBD elements (Page 967)

Adding additional inputs and outputs to FBD elements

Introduction

You can expand several FBD elements with additional inputs that execute arithmetic or binary operations. Such elements are, for example, the instructions "Add" (ADD), "Multiply" (MUL), AND or OR. You can expand the "MOVE value" (MOVE) and "Demultiplex" (DEMUX) instruction boxes by adding additional outputs.

The name of the new inputs and outputs is comprised of the type of inserted element and a consecutive number. The name of a new input is may be "IN2"; the name of a new output may be "OUT2".

Requirements

An FBD element is available that permits the insertion of additional inputs and outputs.

Inserting an additional input

To add an additional input to the box of an FBD element, follow these steps:

1. Right-click on an existing input of the FBD element.
2. Select "Insert input" in the shortcut menu.

   An additional input is added to the box of the FBD element.

Or:

1. Click on the yellow star symbol beside the last input in the instruction box.

   An additional input is added to the box of the FBD element.

Inserting an additional output

To add an additional output to the box of an FBD element, follow these steps:

1. Right-click on an existing output of the FBD element.
2. Select "Insert output" from the shortcut menu.

   An additional output is added to the box of the FBD element.

Or:

1. Click on the yellow star symbol beside the last output of the instruction box.

   An additional output is added to the box of the FBD element.

See also

Selecting FBD elements (Page 962)

Copying FBD elements (Page 963)

Cutting FBD elements (Page 963)

Pasting an FBD element from the clipboard (Page 964)

Replacing FBD elements (Page 964)

Removing instruction inputs and outputs (Page 966)

Deleting FBD elements (Page 967)

Removing instruction inputs and outputs

Introduction

Inputs and outputs which you have added to an instruction can be removed.

Requirement

An FBD element is available, which you have expanded with additional inputs or outputs.

Remove input

To remove an input, follow these steps:

1. Select the input that you want to remove.
2. Select the "Delete" command in the shortcut menu.

   The input of the FBD element is removed.

Remove output

To remove an output, follow these steps:

1. Select the output that you want to remove.
2. Select the "Delete" command in the shortcut menu.

   The output of the FBD element will be removed.

See also

Selecting FBD elements (Page 962)

Copying FBD elements (Page 963)

Cutting FBD elements (Page 963)

Pasting an FBD element from the clipboard (Page 964)

Replacing FBD elements (Page 964)

Adding additional inputs and outputs to FBD elements (Page 965)

Deleting FBD elements (Page 967)

Deleting FBD elements

Requirement

An FBD element is available.

Procedure

To delete an FBD element, follow these steps:

1. Right-click the FBD element that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Selecting FBD elements (Page 962)

Copying FBD elements (Page 963)

Cutting FBD elements (Page 963)

Pasting an FBD element from the clipboard (Page 964)

Replacing FBD elements (Page 964)

Adding additional inputs and outputs to FBD elements (Page 965)

Removing instruction inputs and outputs (Page 966)

Inserting operands in FBD instructions

Inserting operands

The character strings "<???>", "<??.?>" and "..." are inserted as placeholders for the parameters when a FBD element is inserted. The "<???>" and "<??.?>" strings displayed in red indicate parameters that need to be connected. The "..." string displayed in black indicates parameters that may be connected. "<??.?>" stands for Boolean placeholders.

Note

To display the available data types in a tooltip, move the cursor over the placeholder.

Requirement

An FBD element is available.

Procedure

To connect the parameters of an FBD element, follow these steps:

1. Click the placeholder of the parameter.

   An input field is opened.

2. Enter the corresponding parameters, for example a PLC tag, a local tag or a constant.

   ### Note

   If you enter the absolute address of a parameter that has already been defined, this absolute address will be changed to the symbolic name of the parameter as soon as the input is confirmed. If you have not yet defined the parameter, a new tag with this absolute address and the default name "Tag_1" will be entered in the PLC tag table. When you confirm your input, the absolute address will be replaced with the symbolic name "Tag_1".

3. Confirm the parameter with the Enter key.
4. If you have not yet defined the parameter, you can define it directly in the program editor using the shortcut menu.

   See also: "Declaring PLC tags in the program editor (Page 832)".

Or drag from it the PLC tag table:

1. In the project tree, select the "PLC tags" folder and open the PLC tag table.
2. If you have opened the PLC tag table, drag the desired tag to the corresponding location in your program. If you have not opened the PLC tag table yet, open the detail view now. Drag the desired tag from the detail view to the appropriate place in your program.

Or drag from it the block interface:

1. Open the block interface.
2. Drag the desired operand from the block interface to the corresponding location in your program.

Result

- If the syntax is error-free, the displayed parameter is black.
- If there is an error in the syntax, the cursor stays in the input field and a corresponding error message is displayed in the inspector window in the "Info > Syntax" register.

Wiring hidden parameters

Introduction

Depending on the CPU used, you can use complex instructions in your program that are dispatched with the TIA portal. These instructions can contain parameters that are declared as hidden.

If an instruction contains hidden parameters, the instruction box has a small arrow on the lower edge. You can recognize hidden parameters by their white font.

You can show and wire the hidden parameters at any time.

Showing or hiding hidden parameters

To show or hide hidden parameters, follow these steps:

1. Click on the down arrow at the bottom edge of the instruction box to show hidden parameters.
2. Click on the up arrow at the bottom edge of the instruction box to hide hidden parameters.

Wiring hidden parameters

To wire parameters, follow these steps:

1. Wire the hidden parameters like normally visible parameters.

   The hidden parameter is transformed into a visible parameter.

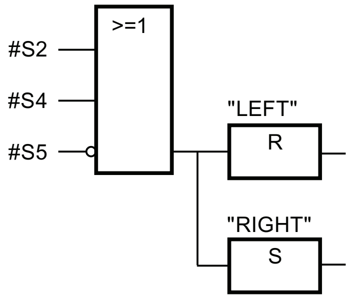
Branches in FBD

Basic information on branches in FBD

Definition

You can use the Function Block Diagram (FBD) programming language to program parallel branches. This is done using branches that are inserted between the boxes. You can insert additional boxes within the branch and in this way build up complex function block diagrams.

The figure below shows an example of the use of branches:



See also

Rules for branches in FBD (Page 970)

Inserting branches in FBD networks (Page 971)

Deleting branches in FBD networks (Page 971)

Rules for branches in FBD

Rules

The following rules apply to the use of branches in FBD:

- Branches are opened downward.
- Branches can be inserted only between FBD elements.
- To delete a branch, you must delete all FBD elements, including the branch itself.
- If you delete the connection between two branches, the FBD elements of the interrupted branch will be positioned freely in the network.

See also

Basic information on branches in FBD (Page 970)

Inserting branches in FBD networks (Page 971)

Deleting branches in FBD networks (Page 971)

Inserting branches in FBD networks

Requirement

A network is available.

Procedure

To insert a new branch in a network, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Branch" in the "Basic instructions" palette.
3. Drag the element from the "Elements" pane to the a required location on a connection line between two boxes.

See also

Rules for branches in FBD (Page 970)

Basic information on branches in FBD (Page 970)

Deleting branches in FBD networks (Page 971)

Deleting branches in FBD networks

Requirement

A branch is available.

Procedure

To delete a branch, follow these steps:

1. Select the connection line that links the branch to the main branch.
2. Select the "Delete" command in the shortcut menu.

Result

The branch is now deleted. Boxes connected to the deleted branch are placed freely within the network.

See also

Rules for branches in FBD (Page 970)

Basic information on branches in FBD (Page 970)

Inserting branches in FBD networks (Page 971)

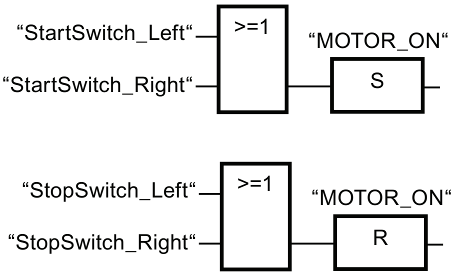
Logic paths in FBD

Basic information on logic paths in FBD

Use of logic paths

The user program will be mapped in one or more networks. The networks can contain one or more logic paths on which the binary signals are arranged in the form of boxes.

The following figure shows an example of the use of several logic paths within a network:



Rules

Remember the following rules when using logic paths:

- Connections are not permitted between logic paths.
- Only one jump instruction is permissible per network. The positioning rules for jump instructions remain valid.

Executing logic paths

Logic paths are executed from top to bottom and from left to right. This means that the first instruction in the first logic path of the first network is executed first. All instructions of this logic path are then executed. After this come all other logic paths of the first network. The next network is executed only after all logic paths have first been executed.

When jumps are used the regular execution of the logic paths is circumvented and the instruction is executed at the jump destination.

Differences between branches and logic paths

The difference between branches and logic paths is that the logic paths are independent branches that can also stand in a different network. Branches, on the other hand, permit the programming of a parallel connection and have a common preceding logic operation.

See also

Inserting a logic path (Page 973)

Deleting a logic operation path (Page 973)

Inserting a logic path

Requirement

- A block is open.
- A network is available.

Procedure

To insert a new logic path in a network, follow these steps:

1. Insert any instruction in a network in such a way that it has no connection to existing instructions.

   A new logic path is inserted.
2. Insert an assignment at the end of the new logic path.
3. Insert additional instructions in the new logic path.

See also

Basic information on logic paths in FBD (Page 972)

Deleting a logic operation path (Page 973)

Deleting a logic operation path

Requirement

A logic path is available.

Procedure

To delete a logic path, proceed as follows:

1. Hold down the left mouse button and draw a frame around the logic path. At the same time, make sure that you select all instructions of the logic path. Alternatively, you can hold down the <Shift> key and select the first the last instruction of the logic path.
2. Right-click on one of the instructions in the logic path.
3. Select the "Delete" command in the shortcut menu.

See also

Basic information on logic paths in FBD (Page 972)

Inserting a logic path (Page 973)

Creating SCL programs

Basics of SCL

Programming language SCL

Programming language SCL

SCL (Structured Control Language) is a high-level programming language based on PASCAL. The language is based on DIN EN 61131-3 (international IEC 1131-3).

The standard standardizes programming languages for programmable logic controllers. The SCL programming language fulfills the PLCopen Basis Level of ST language (Structured Text) defined in this standard.

Language elements

SCL also contains higher programming languages in addition to the typical elements of the PLC, such as inputs, outputs, timers or memory bits.

- Expressions
- Value assignments
- Operators

Program control

SCL provides convenient instructions for controlling the program allowing you, for example, to create program branches, loops or jumps.

Application

SCL is therefore particularly suitable for the following areas of application:

- Data management
- Process optimization
- Recipe management
- Mathematical / statistical tasks

Expressions

Description

Expressions are calculated during the runtime of the program and return a value. An expression consists of operands (such as constants, tags or function calls) and optionally out of operators (such as *, /, + or -). Expressions can be linked together or nested within each other by operators.

Evaluation order

The evaluation of the expression occurs in a specific order that is defined by the following factors:

- Priority of the operators involved
- Left-to-right order
- Brackets

Types of expressions

The following expression types are available depending on the operator:

- Arithmetic expressions

  Arithmetic expressions consist of either a numerical value or combine two values or expressions with arithmetic operators.
- Relational expressions

  Relational expressions compare the values of two operands and yield a Boolean value. The result is TRUE if the comparison is true, and FALSE if it is not met.
- Logical expressions

  Logical expressions combine two operands with logical operators (AND, OR, XOR) or negating operands (NOT).

How expressions are used

You can use the result of an expression in different ways:

- As a value assignment for a tag
- As as a condition for a control instruction
- As a parameter for a calling a block or instruction

See also

Operators and operator precedence (Page 980)

Arithmetic expressions (Page 976)

Relational expressions (Page 978)

Logical expressions (Page 979)

Arithmetic expressions

Description

Arithmetic expressions consist of either a numerical value or combine two values or expressions with arithmetic operators.

Arithmetic operators can process the data types that are allowed in the CPU in use. The data type of the result is determined by the type of the most significant operand. For example, when you link an INT type operand to a DINT type operand, the result is type DINT.

Data types of arithmetic expressions

The following table shows the data types you can use in arithmetic expressions:

Operation	Operator	1. Operand	2. Operand	Result
Power	**	Integer/floating-point number	Integer/floating-point number	Floating-point number
Unary plus	+	Integer/floating-point number TIME	-	Integer/floating-point number TIME
Unary minus	-	Integer/floating-point number TIME	-	Integer/floating-point number TIME
Multiplication	*	Integer/floating-point number	Integer/floating-point number	Integer/floating-point number
		TIME	Integer	TIME
Division	/	Integer/floating-point number	Integer/floating-point number (not equal 0)	Integer/floating-point number
		TIME	Integer	TIME

Operation	Operator	1. Operand	2. Operand	Result
Modulo function	MOD	Integer	Integer	Integer
Addition	+	Integer/floating-point number	Integer/floating-point number	Integer/floating-point number
		TIME	TIME	TIME
		TOD	TIME	TOD
		DT	TIME	DT
Subtraction	-	Integer/floating-point number	Integer/floating-point number	Integer/floating-point number
		TIME	TIME	TIME
		TOD	TIME	TOD
		DATE	DATE	TIME
		TOD	TOD	TIME
		DT	TIME	DT
		DT	DT	Time

For additional information on valid data types, refer to "See also".

Example

The following example shows an arithmetic expression:

```
SCL
"MyTag1":= "MyTag2" * "MyTag3";
```

See also

Expressions (Page 975)

Relational expressions

Description

Relational expressions compare the values of two operands and yield a Boolean value. The result is TRUE if the comparison is true, and FALSE if it is not met.

Relational operators can process the data types that are allowed in the CPU in use. The data type of the result always is BOOL.

Note the following rules when forming relational expressions:

- All tags are comparable within the following data type groups:
  - Integers/floating-point numbers
  - Binary numbers/bit strings
  - String
- With the following data types/data groups, only tags of the same type can be compared:
  - TIME
  - Date and time
- The comparison of strings is based on the ASCII character set. The length of the tags and the numerical value of each ASCII character are used for the comparison.
- S5-TIME tags are not permitted as relational operands. An explicit conversion from S5TIME to TIME is necessary.

Data types of relational expressions

The following table shows the data types/data type groups you can use in relational expressions:

Operation	Operator	1. Operand	2. Operand	Result
Compare for equal, not equal	=, <>	Integer/floating-point number	Integer/floating-point number	BOOL
		Binary number	Binary number	BOOL
		String	String	BOOL
		TIME	TIME	BOOL
		Date and time	Date and time	BOOL
Compare for less than, less than-equal to, greater than, greater than or equal to	<, <=, >, >=	Integer/floating-point number	Integer/floating-point number	BOOL
		String	String	BOOL
		TIME	TIME	BOOL
		Date and time	Date and time	BOOL

For additional information on valid data types, refer to "See also".

Example

The following example shows a relational expression:

SCL

```
IF a > b THEN c:= a;
IF A > 20 AND B < 20 THEN C:= TRUE;
IF A<>(B AND C) THEN C:= FALSE;
```

Note

The comparison for STRING and DT are executed internally in the S7-300/400 by extended instructions. The following operands are not permitted for these functions:

- Parameter of a FC
- In-out parameter of an FB of type STRUCT or ARRAY

See also

Expressions (Page 975)

Logical expressions

Description

Logical expressions combine two operands with logical operators (AND, OR, XOR) or negating operands (NOT).

Logical operators can process the data types that are allowed in the CPU in use. The result of a logical expression is of BOOL data type, if both operands are of BOOL data type. If at least one of both operands is a bit string, then the result is also a bit string and is determined by the type of the highest operand. For example, when you link a BOOL type operand to a WORD type operand, the result is type WORD.

Data types of logical expressions

The following table shows the data types you can use in logical expressions:

Operation	Operator	1. Operand	2. Operand	Result
Negation	NOT	Binary number	-	Binary number
AND logic operation	AND or &	Binary number	Binary number	Binary number
		Bit string	Bit string	Bit string
OR logic operation	OR	Binary number	Binary number	Binary number
		Bit string	Bit string	Bit string
EXCLUSIVE OR logic operation	XOR	Binary number	Binary number	Binary number
		Bit string	Bit string	Bit string

Example

The following example shows a logical expression:

```
SCL
IF "MyTag1" AND NOT "MyTag2" THEN c:=a;
MyTag:=ALPHA OR BETA;
```

See also

Expressions (Page 975)

Operators and operator precedence

Operators and their order of evaluation

Expressions can be linked together or nested within each other by operators.

The order of evaluation for expressions depends on the precedence of operators and brackets. The following basic rules apply:

- Arithmetic operators are evaluated before relational operators and relational operators are evaluated before logical operators.
- Operators with no precedence are evaluated according to their occurrence from left to right.
- Operations in brackets are evaluated first.

The following table provides an overview of the operators and their precedence:

Operator	Operation	Precedence
Arithmetic expressions		
**	Power	2
+	Unary plus	3

Operator	Operation	Precedence
-	Unary minus	3
*	Multiplication	4
/	Division	4
MOD	Modulo function	4
+	Addition	5
-	Subtraction	5
Relational expressions		
<	Less than	6
>	Greater than	6
<=	Less than or equal	6
>=	Greater than or equal	6
=	Equal	7
<>	Not equal	7
Logical expressions		
NOT	Negation	3
AND or &	Boolean AND	8
XOR	Exclusive OR	9
OR	Boolean OR	10
Miscellaneous operations		
( )	Brackets	1
:=	Assignment	11

See also

Expressions (Page 975)

Value assignments

Definition

You can use a value assignment to assign the value of an expression to a tag. On the left side of the assignment is the tag that takes the value of the expression on the right.

The name of a function can also be specified as an expression. The function is called by the value assignment and sends its return value back to the tag on the left.

The data type of value assignment is defined by the data type of the tag on the left. The data type of the expression on the right must match this type.

For additional information on compatibility and conversion of data types, refer to "See also".

Value assignments for STRUCT data type or PLC data type

An entire structure can be assigned to another if the structures are identically organized and the data types as well as the names of the structural components match.

You can assign a tag, an expression or another structural element to an individual structural element.

Value assignments for the ARRAY data type

An entire ARRAY can be assigned to another ARRAY if both the data types of the ARRAY elements as well as the ARRAY limits match.

You can assign a tag, an expression or another ARRAY element to an individual ARRAY element.

Value assignments for the STRING data type

An entire STRING can be assigned to another STRING. If the assigned character string is longer than the string on the left, a warning is generated during compiling.

You can assign another STRING element to an individual STRING element.

Examples

The following table shows examples for value assignments:

SCL	
`"MyTag1" := "MyTag2";`	`(* Assignment of a tag*)`
`"MyTag1" := "MyTag2" * "MyTag3";`	`(* Assignment of an expression*)`
`"MyTag" := "MyFC"();`	`(* Call a function, which assigns its return value to the "MyTag" tag*)`
`#MyStruct.MyStructElement := "MyTag";`	`(* Assignment of a tag to a structure element*)`
`#MyArray[2] := "MyTag";`	`(* Assignment of a tag to an ARRAY element*)`
`"MyTag" := #MyArray[1,4];`	`(* Assignment of an ARRAY element to a tag*)`
`#MyString[2] := #MyOtherString[5];`	`(* Assignment of a STRING element to another STRING element*)`

Settings for SCL

Overview of the settings for SCL

Overview

The following tables show the settings you can make for SCL:

Editor settings

Group	Setting	Description
View	Keyword highlighting	Notation used to represent the keywords of the programming language. You can choose between uppercase and lowercase letters or a notation corresponding to the conventions of the Pascal programming language.

Default settings for new blocks

If you create new blocks, the following settings are set as default values. You can change these in the block properties at a later point in time.

Group	Setting	Description
Compile	Create extended status information	Allows all tags in a block to be monitored. The memory requirements of the program and execution times increase, however, with this option.
	Check ARRAY limits	Checks at runtime whether array indices are within the declared range for an ARRAY. If an array index exceeds the permissible range, the enable output ENO of the block is set to "0".
	Set ENO automatically	Checks at runtime whether errors occur in the processing of certain instructions. If a runtime error occurs, the enable output ENO of the block is set to "0".

See also

Changing the settings (Page 983)

Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.

   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the settings for SCL (Page 982)

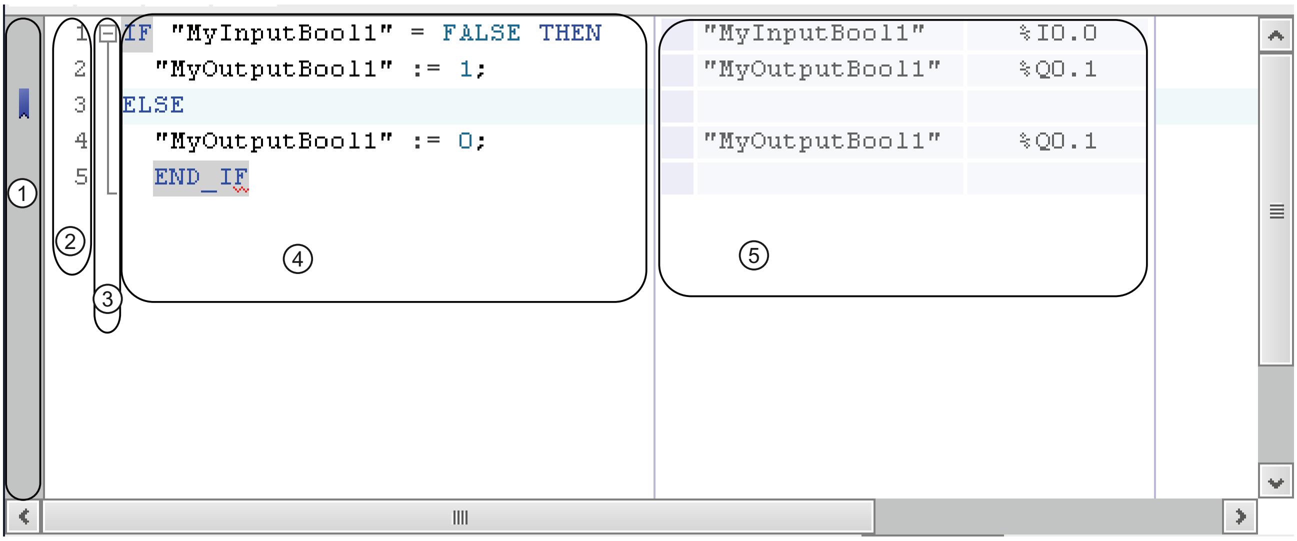
The programming window of SCL

Overview of the programming window

Function

The programming window is the work area, where you enter the SCL program.

The following figure shows the programming window of SCL:



```
IF "MyInputBool1" = FALSE THEN
  "MyOutputBool1" := 1;
ELSE
  "MyOutputBool1" := 0;
  END_IF
```

```
"MyInputBool1"     %I0.0
"MyOutputBool1"    %Q0.1

"MyOutputBool1"    %Q0.1
```

The programming window consists of the following sections:

Section	Meaning
① Sidebar	You can set bookmarks and breakpoints in the sidebar.
② Line numbers	The line numbers are displayed to the left of the program code.
③ Outline view	The outline view highlights related code sections.
④ Code area	You edit the SCL program in the code area.
⑤ Display of the absolute operands	This table shows the assignment of symbolic operands to absolute addresses.

See also

Using bookmarks (Page 988)

Customizing the programming window (Page 985)

Indenting and outdenting lines (Page 986)

Expanding and collapsing sections of code (Page 987)

Customizing the programming window

Introduction

You can customize the appearance of the programming window and the program code in the following way:

- By setting the font, size and color
- By setting the tab spacing
- By displaying the line numbers
- By showing or hiding the absolute operands

Setting the font, size and color

To set the font, size and color, follow these steps:

1. Select the "Settings" command in the "Options" menu.

   The "Settings" window is displayed in the work area.
2. Select the "General > Text editors" group.
3. Select the desired font and font size or choose a font color for the individual language elements.

Setting the tab spacing

To provide a better overview of the program, lines are indented according to syntax. Define the depth of indentation with the tab spacing.

To set the tab spacing, follow these steps:

1. Select the "Settings" command in the "Options" menu.

   The "Settings" window is displayed in the work area.
2. Select the "General > Text Editors" group.
3. Set the tab spacing.

Show line numbers

To display the line numbers, follow these steps:

1. Select the "Settings" command in the "Options" menu.

   The "Settings" window is displayed in the work area.
2. Select the "General > Text editors" group.
3. Select the "Show line numbers" option.

Show or hide the absolute operands

You can show the assignment of symbolic and absolute operands in a table next to the program code, if required.

To hide or show the display of the absolute operands, follow these steps:

1. Click the "Absolute/symbolic operands" icon in the toolbar.

   The display of the absolute operands appears.
2. To move the display, click the table and drag it to the desired position while holding down the mouse button.
3. To change the width of the table, click on the right or left table border and drag it to the right or left while holding down the mouse button.

See also

Using bookmarks (Page 988)

Overview of the programming window (Page 984)

Indenting and outdenting lines (Page 986)

Expanding and collapsing sections of code (Page 987)

Indenting and outdenting lines

Introduction

To provide a better overview of the program, lines are indented according to syntax. However, you can also manually indent individual lines.

Procedure

To indent or outdent individual lines, follow these steps:

1. Press the "Indent text", "Outdent text" button into the toolbar of the programming editor.

Note

You can set the width of the indent in "Options > Settings".

See also

Using bookmarks (Page 988)

Overview of the programming window (Page 984)

Customizing the programming window (Page 985)

Expanding and collapsing sections of code (Page 987)

Expanding and collapsing sections of code

Introduction

SCL instructions can span several lines. Examples for this are program control instructions or block calls.

These instructions are identified as follows:

- An outline view between the display line number and the program code marks the entire code section.
- When you select the opening keyword, the closing keyword is automatically highlighted.

Procedure

To expand or collapse the code section, follow these steps:

1. Click the minus sign in the outline view.

   The code section closes.
2. Click the plus sign in the outline view.

   The code section opens.

See also

Using bookmarks (Page 988)

Overview of the programming window (Page 984)

Customizing the programming window (Page 985)

Indenting and outdenting lines (Page 986)

Using bookmarks

Basics of bookmarks

Function

You can use bookmarks to mark program locations in extensive programs so that you can find them quickly later if they need revising. Bookmarks are displayed in the sidebar of the programming window. You can navigate between multiple bookmarks within a block using menu commands.

Bookmarks are saved with the project and are therefore available for anyone who wants to edit the block. However, they are not loaded to a device.

Bookmarks are not evaluated when blocks are compared.

See also

Setting bookmarks (Page 988)

Navigating between bookmarks (Page 989)

Deleting bookmarks (Page 989)

Setting bookmarks

Requirement

The SCL block is open.

Procedure

To set a bookmark, follow these steps:

1. Click on the desired line in the sidebar.
2. Select the "Bookmarks > Set" command in the shortcut menu.

Or:

1. Click on the desired line in the sidebar.
2. Click the "Set/delete bookmark" button in the toolbar.

Or:

1. Hold down the <Ctrl> key.
2. Click on the line in the sidebar in which you want to place the bookmark.

Result

A bookmark is placed in the program code.

See also

Basics of bookmarks (Page 988)

Navigating between bookmarks (Page 989)

Deleting bookmarks (Page 989)

Navigating between bookmarks

Requirement

Several bookmarks are set in a block.

Procedure

To navigate between bookmarks, follow these steps:

1. Set the insertion cursor in the program code.
2. In the "Edit" menu, select the "Go to > Next bookmark" or "Go to > Previous bookmark" command.

Or:

1. Set the insertion cursor in the program code.
2. In the toolbar of the programming editor, click the "Go to next bookmark", "Go to previous bookmark" button.

Or:

1. Click in the sidebar.
2. Select the "Bookmarks > Next" or "Bookmarks > Previous" command in the shortcut menu.

Result

The line with the bookmark is highlighted.

See also

Basics of bookmarks (Page 988)

Setting bookmarks (Page 988)

Deleting bookmarks (Page 989)

Deleting bookmarks

You can delete individual bookmarks or all bookmarks from the block or the CPU.

Deleting individual bookmarks

To delete an individual bookmark, follow these steps:

1. Click on the desired line in the sidebar.
2. Select the "Bookmarks > Remove" command in the shortcut menu.

Or:

1. Click on the desired line in the sidebar.
2. In the "Edit" menu, select the "Bookmarks > Remove" command.

Or:

1. Click on the desired line in the sidebar.
2. Click the "Set/delete bookmark" button in the toolbar.

Deleting all bookmarks from the block

To delete all bookmarks from the block, follow these steps:

1. Click in the sidebar.
2. Select the "Bookmarks > Delete all from block" command in the shortcut menu.

Or:

1. In the "Edit" menu, select the "Bookmarks > Delete all from block" command.

See also

Basics of bookmarks (Page 988)

Setting bookmarks (Page 988)

Navigating between bookmarks (Page 989)

Entering SCL instructions

Rules for SCL instructions

Instructions in SCL

SCL recognizes the following types of instructions:

- Value assignments

  Value assignments are used to assign a tag a constant value, the result of an expression or the value of another tag.
- Instructions for program control

  Instructions for program control are used to implement program branches, loops or jumps.
- Additional instructions from the "Instructions" task card

  The "Instructions" task card offers a wide selection of standard instructions that you can use in your SCL program.
- Block calls

  Block calls are used to call up subroutines that have been placed in other blocks and to further process their results.

Rules

You need to observe the following rules when entering SCL instructions:

- Instructions can span several lines.
- Each instruction ends with a semicolon (;).
- No distinction is made between upper and lower case.
- Comments serve only for documentation of the program. They do not affect the program execution.

Examples

The following examples shows the various types of instructions:

SCL

```
// Example of a value assignment
"MyTag":= 0 ;
// Example of a block call
"MyD"B."MyFB" (ParamInput:= 10) ;
// Example for a program control instruction
WHILE "Counter" < 10 DO
 "MyTAG" := "MyTag" + 2;
END_WHILE;
```

See also

Basics of SCL (Page 974)

Entering SCL instructions manually

Requirement

An SCL block is open.

Procedure

To enter SCL instructions, follow these steps:

1. Enter the syntax of the instruction using the keyboard.

   You are supported by the auto-complete function when performing this task. It offers all the instructions and operands that are allowed at the current location.
2. Select the required instruction or the desired operand from the auto-complete function.

   If you select an instruction that requires specification of operands, placeholders for the operands are inserted into the program. The placeholders for the operands are highlighted in yellow. The first placeholder is selected.
3. Replace this placeholder with an operand.
4. Use the <TAB> key to navigate to all other placeholders and replace them with operands.

Note

You can also drag-and-drop a defined operand from the PLC tag table or from the block interface into the program.

Result

The instruction is inserted.

The programming editor performs a syntax check. Incorrect entries are displayed in red and italics. In addition, you also receive a detailed error message in the inspector window.

Inserting SCL instructions using the "Instructions" task card

The "Instructions" task card offers a wide selection of instructions that you can use in your SCL program. The SCL-specific instructions for program control are available in the "Instructions" task card.

Requirement

An SCL block is open.

Procedure

To insert SCL instructions into a program using the "Instructions" task card, follow these steps:

1. Open the "Instructions" task card.
2. To insert the instruction, select one of the following steps:
   - Navigate to the SCL instruction you want to insert and drag-and-drop it to the required line in the program code. The insertion location is highlighted by a green rectangle.
   - Select the location in the program code where you want to insert the instruction and then double-click on the instruction you want to insert.

   The instruction is inserted in the program. The placeholders for the operands are highlighted in yellow. The first placeholder is selected.
3. Replace this placeholder with an operand. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
4. Use the <TAB> key to navigate to all other placeholders and replace them with operands.

Result

The instruction is inserted.

The programming editor performs a syntax check. Incorrect entries are displayed in red and italics. In addition, you also receive a detailed error message in the inspector window.

Using Favorites in SCL

Adding SCL instructions to the Favorites

Requirement

- A block is open.
- The multipane mode is set for the "Instructions" task card or the Favorites are also displayed in the editor.

Procedure

To add SCL instructions to the Favorites, follow these steps:

1. Open the "Instructions" task card.
2. Maximize the "Basic instructions" pane.
3. Navigate in the "Basic instructions" pane to the instruction that you want to add to the Favorites.
4. Drag-and-drop the instruction into the "Favorites" pane or into the Favorites area in the program editor.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Inserting SCL instructions using Favorites (Page 994)

Removing SCL instructions from the Favorites (Page 995)

Inserting SCL instructions using Favorites

Requirement

- A block is open.
- Favorites are available.

Procedure

To insert an instruction into a program using Favorites, follow these steps:

1. Drag-and-drop the desired instruction from Favorites to the desired position.

Or:

1. Select the position in the program where you want to insert the instruction.
2. In the Favorites, click on the instruction you want to insert.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Adding SCL instructions to the Favorites (Page 993)

Removing SCL instructions from the Favorites (Page 995)

Removing SCL instructions from the Favorites

Requirement

A code block is open.

Procedure

To remove instructions from Favorites, follow these steps:

1. Right-click on the instruction you want to remove.
2. Select the "Remove instruction" command in the shortcut menu.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Adding SCL instructions to the Favorites (Page 993)

Inserting SCL instructions using Favorites (Page 994)

Insert block calls in SCL

Basic information on the block call in SCL

Calling function blocks

Syntax of a call

The following syntax is used to call a function block as a single or multi-instance:

```
<DB name>.<FB name> (Parameter list) //Call as a single instance
<#Instance name> (Parameter list) // Call as multi-instance
```

Calling as single instance or multi-instance

Function blocks can be called either as a single instance or a multi-instance.

- Calling as a single instance

  The called function block stores its data in a data block of its own.

- Calling as a multi-instance

  The called function block stores its data in the instance data block of the calling function block.

For additional information on the types of calls, refer to "See also".

Parameter list

If you call another code block from a SCL block, you can supply the formal parameters of the called block with actual parameters.

The specification of the parameters has the form of a value assignment. This value assignment enables you to assign values (actual parameters) to the parameters you have defined in the called block.

The formal parameters of the called code block are listed in brackets directly after the call. Input and in-out parameters have the assignment identifier ":=", output parameters have the assignment identifier "=>". A gray placeholder placed after the parameter shows the required data type and the type of the parameter.

Rules for supplying parameters

The following rules apply to supplying parameters:

- Constants, tags and expressions can be used as actual parameters.
- The assignment order is not of importance.
- The data types of formal and actual parameters must match.
- The individual assignments are separated by commas.
- If the called block has only one parameter, it is sufficient to specify the actual parameter in the brackets. The formal parameter need not be specified.

See also

Manually inserting block calls (Page 1000)

Inserting block calls with drag-and-drop (Page 1001)

Examples for calling a function block in SCL (Page 998)

Calling functions

Syntax of a call

The following syntax is used to call a function:

```
<Function name> (Parameter list);     //Standard call
<Operand>:=<Function name> (Parameter list); // Call in an
expression
```

Return value

The call options of functions depend on whether the function returns a return value to the calling block. The return value is defined at the RET_VAL parameter. If the RET_VAL parameter is of the VOID data type, then the function will not return a value to the calling block. If the RET_VAL parameter has another data type, then the function returns a return value of this data type.

All data types are permitted in SCL for the RET_VAL parameter except ANY, ARRAY and STRUCT as well as the parameter types TIMER and COUNTER.

Call options

There are two possibilities for calling functions in SCL:

- Standard call for functions with and without a return value

  With a standard call, the results of the function is made available as an output and in-out parameter.
- Call in an expression for functions with a return value

  Functions that return a return value can be used in any expression in place of an operand, for example, in a value assignment.

  The function calculates the return value, which has the same name as the function and returns it to the calling block. There the value replaces the function call.

  After the call, the results of the function will be available as return value or as an output and in-out parameter.

Parameter list

If you call another code block from a SCL block, you need to supply the formal parameters of the called block with actual parameters.

The specification of the parameters has the form of a value assignment. This value assignment enables you to assign values (actual parameters) to the parameters you have defined in the called block.

The formal parameters of the called code block are listed in brackets directly after the call. Input and in-out parameters have the assignment identifier ":=", output parameters have the assignment identifier "=>". A gray placeholder placed after the parameter shows the required data type and the type of the parameter.

Rules for supplying parameters

The following rules apply to supplying parameters to functions:

- All parameters of the function must be supplied.
- The assignment order is not of importance.
- Constants, tags and expressions can be used as actual parameters.
- The data types of formal and actual parameters must match.
- The individual assignments are separated by commas.
- If the called block has only one parameter, it is sufficient to specify the actual parameter in the brackets. The formal parameter need not be specified.
- When you call functions in SCL, you cannot use the release mechanism via EN. Use an IF statement instead to call functions conditionally.

See also

Manually inserting block calls (Page 1000)

Inserting block calls with drag-and-drop (Page 1001)

Examples for calling functions in SCL (Page 999)

Examples for calling a function block in SCL

Calling as a single instance

The following example shows the call of an FB as a single instance:

SCL

```
// Call as a single instance
"MyDB"."MyFB" (MyInput:=10, MyInout:= "Tag1");
```

Result

After the "MyFB" block is executed, the value determined for the "MyInout" in-out parameter is made available in Tag1 in the "MyDB" data block.

Calling as a multi-instance

The following example shows the call of an FB as a multi-instance:

SCL

```
// Call as a multi-instance
#MyInstance."MyFB" (MyInput:= 10, MyInout:= "Tag1") ;
```

Result

After the "MyFB" block is executed, the value determined for the "MyInout" in-out parameter is made available in "Tag1" in the data block of the calling code block.

See also

Calling function blocks (Page 995)

Manually inserting block calls (Page 1000)

Inserting block calls with drag-and-drop (Page 1001)

Examples for calling functions in SCL

Standard call

The following example shows a standard function call:

SCL

```
// Standard function call
"MyFC" (MyInput:=10, "MyInOut":= "Tag1");
```

Result

After the "MyFC" block is executed, the value determined for the "MyInOut" in-out parameter is made available in "Tag1" in the "MyFB_DB". The "Tag1" operand must be further processed in the calling block.

Call in a value assignment

The following example shows a function call in a value assignment:

SCL

```
(*Call in a value assignment, a return value was defined for "MyFC" *)
#MyOperand := "MyFC" (MyInput1:=3, MyInput2:=2, MyInput3:=8.9, MyInOut:= "Tag1") ;
```

Result

The return value will be transferred from "MyFC" to "MyOperand".

Call in an arithmetic expression

The following example shows a function call in an arithmetic expression:

```
SCL
(*Call in a mathematical expression, a return value was defined for "MyFC" *)
#MyOperand := "Tag2" + "MyFC" (MyInput1:=3, MyInput2:=2, MyInput3:=8.9)
```

Result

The return value of "MyFC" will be added to "Tag2" and the result will be transferred to "MyOperand".

See also

Calling functions (Page 997)

Manually inserting block calls (Page 1000)

Inserting block calls with drag-and-drop (Page 1001)

Manually inserting block calls

You can insert calls for functions (FCs) and function blocks (FBs).

Inserting a call for a function (FC)

Proceed as follows to insert a function call:

1. Enter the function name.
2. Confirm your entry with the Return key.

   The syntax for the function call including the parameter list is added to the SCL program. The placeholders for the actual parameters are highlighted in yellow. The first placeholder is selected.
3. Replace this placeholder with an actual parameter. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
4. Use the <TAB> key to navigate to all other placeholders and replace them with actual parameters.

Inserting a call for a function block (FB)

To insert a call for a function block (FB), follow these steps:

1. Enter the name of the function block.
2. Confirm your entry with the Return key.

   The "Call options" dialog opens.
3. In the dialog, specify whether you want to call the block as a single or multi-instance.
   – If you click the "Single instance" button, in the "Name" field enter a name for the data block to be assigned to the call.
   – If you click the "Multi-instance" button, in the "Name in the interface" field enter a name of the tag with which the called function block is to be entered as a static tag in the interface of the calling block.
4. Confirm your entries with "OK".

   The syntax for the function block call including the parameter list is added to the SCL program. The placeholders for the actual parameters are highlighted in yellow. The first placeholder is selected.
5. Replace this placeholder with an actual parameter. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
6. Use the <TAB> key to navigate to all other placeholders and replace them with actual parameters.

Result

The block call is inserted.

If you specify an instance data block that does not exist when calling a function block, it is created.

See also

Updating block calls (Page 1003)

Basic information on the block call in SCL (Page 995)

Inserting block calls with drag-and-drop

You can insert calls for existing functions (FC) and function blocks (FB) using a drag-and-drop operation from the project tree.

Requirement

The function to be called (FC) or the function block (FB) to be called is present.

Inserting a call for a function (FC)

To insert a function call using drag-and-drop, follow these steps:

1. Drag the function from the project tree into the program.

   The syntax for the function call including the parameter list is added to the SCL program. The placeholders for the actual parameters are highlighted in yellow. The first placeholder is selected.
2. Replace this placeholder with an actual parameter. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
3. Use the <TAB> key to navigate to all other placeholders and replace them with actual parameters.

Inserting a call for a function block (FB)

To insert a call for a function block (FB) using drag-and-drop, follow these steps:

1. Drag the function block from the project tree and drop it into the program.

   The "Call options" dialog opens.
2. In the dialog, specify whether you want to call the block as a single or multi-instance.
   - If you click the "Single instance" button, in the "Name" field enter a name for the data block to be assigned to the call.
   - If you click the "Multi-instance" button, in the "Name in the interface" field enter a name of the tag with which the called function block is to be entered as a static tag in the interface of the calling block.
3. Confirm your entries with "OK".

   The syntax for the function block call including the parameter list is added to the SCL program. The placeholders for the actual parameters are highlighted in yellow. The first placeholder is selected.
4. Replace this placeholder with an actual parameter. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
5. Use the <TAB> key to navigate to all other placeholders and replace them with actual parameters.

Result

The block call is inserted.

If you specify an instance data block that does not exist when calling a function block, it is created.

See also

Updating block calls (Page 1003)

Basic information on the block call in SCL (Page 995)

Updating block calls

If interface parameters of a called block are changed, the block call can no longer be executed correctly. You can avoid such inconsistent block calls by updating the block calls.

You have two options to update the block calls:

- Explicit updating in the program editor.

  The inconsistencies within the open block are displayed and can be updated.
- Implicit updating during compilation.

  All block calls in the program as well as the used PLC data types will be updated.

Update blocks in the program editor

To update a block call within a block, follow these steps:

1. Open the block in the program editor.
2. Click "Update inconsistent block calls" in the toolbar.

   Inconsistent calls are displayed.
3. Correct the inconsistencies.

Update block calls during compilation

Proceed as follows to update all block calls and uses of PLC data types during compilation implicitly:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebulid all blocks)" in the shortcut menu.

See also

Manually inserting block calls (Page 1000)

Inserting block calls with drag-and-drop (Page 1001)

Inserting comments

Commenting program code

You have various options for commenting SCL programs:

- Line comment

  A comment line starts with "//" and extends to the end of the line.
- Comment section

  A comment section is introduced with "(* and completed by "*)". It can span several lines.

Inserting line comments

To insert line comments, follow these steps:

1. Type "//" at the position where you want to place the comment. This does not have to be the beginning of the line.
2. Enter the comment.

Inserting a comment section

To insert a comment section, follow these steps:

1. Type "(*" at the position where you want to place the comment. This does not have to be the beginning of the line.
2. Enter the comment.
3. Complete the comment with "*)".

Disabling one or more lines with comments

To disable program code with comments, follow these steps:

1. Select the code lines you want to disable.
2. Click the "Disable code" button in the editor.

   The line beginning "//" is inserted in the selected lines. The code that follows is interpreted as a comment. If lines already containing a line comment are disabled, "//" is inserted as well. If these lines are enabled again, the original comments are retained.

Enabling comment lines

In order to once again enable lines that have been commented out in the code, follow these steps:

1. Select the code lines you want to enable.
2. Click the "Enable code" button in the editor.

   The "//" mark for line comments at the beginning of the line is removed.

Example

The following code contains comment sections and line comments

```
(*************************************************************************************
  A description of the instructions that follow can be placed here
**************************************************************************************)
IF "MyVal1" > 0 THEN //No division by 0
    "MyReal" := "MyVal2" (* input value *) / "MyVal1" (* measured value *);
END_IF;
//Data type conversion
"MyInt" := REAL_TO_INT("MyReal");
```

Editing SCL instructions

Selecting instructions

You can select individual instructions or all instructions of a block.

Requirement

An SCL block is open.

Selecting individual instructions

To select individual instructions, follow these steps:

1. Set the insertion mark before the first character that you want to select.
2. Press and hold down the left mouse button.
3. Move the cursor to a position after the last character that you want to select.
4. Release the left mouse button.

Selecting all the instructions of a program

To select all instructions, follow these steps:

1. In the "Edit" menu, select the "Select All" command or use the keyboard shortcut <Ctrl+A>.

Note

When you select the opening keyword of an instructing, the closing keyword is automatically highlighted.

Copying, cutting and pasting instructions

Copying an instruction

To copy an instruction, follow these steps:

1. Select the instruction you want to copy.
2. Select "Copy" in the shortcut menu.

Cutting an instruction

To cut an instruction, follow these steps:

1. Select the instruction you want to cut.
2. Select the "Cut" command in the shortcut menu.

Inserting an instruction from the clipboard

To insert an instruction from the clipboard, follow these steps:

1. Copy or cut an instruction.
2. Click on the position at which you want to insert the instruction.
3. Select "Paste" in the shortcut menu.

Deleting instructions

Requirement

An SCL block is open.

Procedure

To delete an instruction, follow these steps:

1. Select the instruction you want to delete.
2. Select the "Delete" command in the shortcut menu.

Eliminating syntax errors in the program

Basic information on syntax errors

Syntax errors

Below are some examples of syntax errors:

- Missing separators or the use of too many separators
- Incorrect keyword spelling
- Incorrect jump label spelling/notation
- Notation which does not match the mnemonic set (e.g. "I 2.3" instead of "E 2.3")
- The use of key words as operands

Identification of syntax errors

Syntax errors are underlined in red or appear in red type.

This identification allows you to recognise incorrect inputs at a glance and jump from error to error to eliminate them. Syntax errors are also listed in the "Info" tab of the inspector window with an error message.

See also

Finding syntax errors in the program (Page 1008)

Finding syntax errors in the program

Procedure

To find syntax errors in the program, follow these steps:

1. Select the position in the program in which you wish to look for errors.
2. Click "Go to next error" in the toolbar.

   The first error after the position you have selected will be marked.

You can use "Go to next error" and "Go to previous error" in the toolbar to find and correct all errors in the block.

Or:

1. Open the error list in the inspector window with "Info > Syntax".

   All syntax errors are listed in the table with a short description of the error.
2. If there are any errors, click on the blue question mark next to the error text to obtain information on eliminating the problem.
3. Double-click the error you want to correct.

   The corresponding error is highlighted.

See also

Basic information on syntax errors (Page 1007)

Changing the programming language

Rules for changing the programming language

Rules

Observe the following rules if you want to change the programming language for a block:

- You can only change between the programming languages LAD, FBD and STL.
- You cannot switch blocks programmed in the programming languages SCL or GRAPH. In GRAPH blocks, however, you can change between LAD and FBD as network languages.
- If the language of individual networks of the block cannot be changed, these networks will be displayed in their original language.
- You can only change the programming language of entire blocks. The programming language cannot be changed for individual networks.
- However, you can create networks within a block using another programming language and then copy them into the desired block.

Change the programming language

Procedure

To change the programming language, follow these steps:

1. Right-click the block in the project tree.
2. Select the "Properties" command in the shortcut menu.

   The dialog with the properties of the block opens.
3. Select the "General" entry in the area navigation.
4. Select the new programming language in the "Language" drop-down list.
5. Confirm your selection with "OK".

See also

Rules for changing the programming language (Page 1008)

Handling program execution errors

Basics of error handling

Introduction

Program execution errors are programming or I/O access errors. You have a number of different options for responding to program execution errors depending on the CPU used.

Handling program execution errors in S7-300/400

You can program the program execution error OB (OB 85) for S7-300/400 CPUs. If a program execution error occurs and you do not use the program execution error OB, the CPU will switch to "STOP" mode.

You will find further information about the program execution error OB in the "Configuring devices and networks" references.

Handling program execution errors in S7-1200

You can set error handling yourself in S7-1200. This allows you to specify how the system should respond to any program execution errors that occur. The following options are available:

- You use the following system reactions provided by the operating system:
  - STOP
  - Ignore

  The system reactions are always used for handling programming errors if you do not use local error handling. If you set local error handling for a block, this has priority over the system reaction.
- You use separate local error handling. Local error handling is error handling within a block. Local error handling has the following advantages:
  - The error information is stored in the system memory, which you can query and evaluate.
  - You can use the error information to program a response in the block to the error that has occurred.
  - Programmed error evaluation and error reactions do not interrupt the program cycle.
  - The system performance is not unnecessarily burdened by the local error handling. If no errors occur, programmed error analyses and reactions are not executed.

    Local error handling applies only to blocks for which it has been set explicitly. If local error handling is set for a block, the system reaction is ignored during the execution of this block.

See also

GetError: Get error locally (Page 1535)

GetErrorID: Get error ID locally (Page 1539)

GetError: Get error locally (Page 1363)

GetErrorID: Get error ID locally (Page 1367)

Local error handling

Principles of local error handling

Introduction

Local error handling makes it possible to query the occurrence of errors within a block and evaluate the associated error information. You can set local error handling for organization blocks (OBs), function blocks (FBs), and functions (FCs). If local error handling is enabled, the system reaction is ignored.

Local error handling applies only to blocks for which it has been set explicitly. The local error handling setting is not assumed by a calling block, nor is it transferred to called blocks. For higher-level blocks and lower-level blocks, the system settings still apply provided dedicated error handling has not been programmed for these blocks.

General procedure for local error handling

When errors occur while a block is being executed with local error handling, a predefined response is initiated based on the following error types:

- Write errors: These errors are ignored, and program execution simply continues.
- Read errors: Program execution continues with the substitute value "0".
- Execution errors: Execution of the instruction is aborted. Program execution resumes with the next instruction.

Information about the first error that occurs is stored in the system memory. This information can be queried and output with an instruction (GetError or GetErrorID). Error information is output in a format that can undergo additional processing. You can use additional instructions to analyze error information and program a reaction to the error based.

When information about the first error is queried, the error memory space in the system memory is enabled. Then, when additional errors occur, information about the next error is output.

Instructions for local error handling

You can use the following instructions for local error handling:

- GetError: Get error locally
- GetErrorID: Get error ID locally

The instructions differ in the amount of error information that is output with each one.

For additional information on the instructions, refer to "See also".

See also

GetError: Get error locally (Page 1535)

GetErrorID: Get error ID locally (Page 1367)

GetErrorID: Get error ID locally (Page 1539)

GetError: Get error locally (Page 1363)

Error output priorities

Overview of the priorities

In local error handling, information about the first error that occurred is displayed. If multiple errors occur at the same time while an instruction is being executed, these errors are displayed according to their priority. The following table shows the priority of different types of errors.

Priority	Error type
1	Error in the program code
2	Missing reference
3	Invalid range
4	DB does not exist
5	Operands are not compatible
6	Width of specified area is not sufficient
7	Timers or counters do not exist
8	No write access to a DB
9	I/O error
10	Instruction does not exist
11	Block does not exist
12	Invalid nesting depth

The highest priority is 1 and the lowest priority is 12.

See also

GetError: Get error locally (Page 1535)

GetErrorID: Get error ID locally (Page 1367)

GetErrorID: Get error ID locally (Page 1539)

GetError: Get error locally (Page 1363)

Enabling local error handling for a block

Introduction

Local error handling is enabled for a block if you insert one of the following instructions in a network.

- GetError: Get error locally
- GetErrorID: Get error ID locally

For additional information on the instructions, refer to "See also".

If local error handling is enabled for a block, the system reactions for this block are ignored.

Requirement

- The block is open.
- Die "Instructions" task card is open.

Procedure

To enable local error handling for a block, proceed as follows:

1. Navigate to the "Basic instructions" pane of the "Instructions" task card.
2. Open the "Program Control" folder.
3. Drag the instruction "Get error locally" (GetError) or "Get error ID locally" (GetErrorID) to the required network.

Result

Local error handling is enabled for the open block. In the Inspector window under "Properties > Attributes", the "Handle errors within block" check box is selected. This setting cannot be edited in the Inspector window. The local error handling can be disabled by deleting the inserted instructions to the local error handling.

See also

GetError: Get error locally (Page 1535)

GetErrorID: Get error ID locally (Page 1367)

GetErrorID: Get error ID locally (Page 1539)

GetError: Get error locally (Page 1363)

9.1.4.3 Programming data blocks

Basic principles for programming of data blocks

A data block (DB) is used to save the values that are written during execution of the program.

In contrast to the code block, the data block contains only tag declarations. It contains no networks or instructions. The tag declarations define the structure of the data block.

Types of data blocks

There are two types of data blocks:

- Global data blocks

  The global data block is not assigned to a code block. You can access the values of a global data block from any code block. A global data block contains only static tags.

  The structure of the global data block can be freely defined. In the declaration table for data blocks, you declare the data elements that are to contained in the global data block.

- Instance data blocks

  The instance data block is assigned directly to a function block (FB). The structure of an instance data block cannot be freely defined, but is instead determined by the interface declaration of the function block. The instance data block contains exactly those block parameters and tags that are declared there.

  However, you can define instance-specific values in the instance data block, for example, start values for the declared tags.

PLC data types as a template for global data blocks

PLC data types can be used as templates for the creation of global data blocks with identical data structures. You create the structure as PLC data type only once and then generate the required data blocks by assigning the PLC data type.

System data types as a template for global data blocks

System data types can also be used as templates for creating global data blocks with identical data structure. System data types already have a pre-defined structure. You insert the system data type in the program only once and then generate additional data blocks with an identical structure by assigning the system data type.

Access modes

There are two different modes of accessing data values in data blocks:

- Data blocks with optimized access (only S7-1200)

  Data blocks with optimized access have no fixed defined structure. The declaration elements contain only one symbolic name in the declaration, no fixed addressing within the block. You access the data values in these block via symbolic names.

- Data blocks with standard access (all CPU families)

  Data blocks with standard access have a fixed structure. The declaration elements contain both a symbolic name in the declaration and a fixed address within the block. You can access the data values in these blocks via symbolic names or the address.

Retentivity of data values

To prevent data loss in the event of power failure, you can store the data values in a retentive memory area.

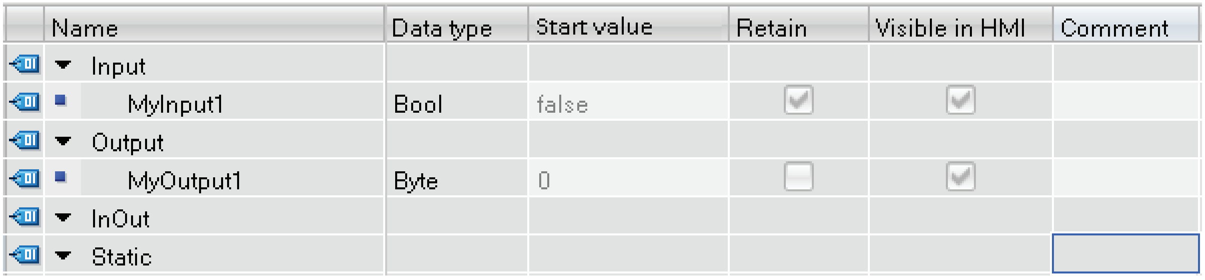
See also

Creating data blocks (Page 846)

Structure of the declaration table for data blocks

Structure of the declaration table for data blocks

The figure below shows the structure of the declaration table for data blocks. The display will vary depending on type of block and type of access.



Name	Data type	Start value	Retain	Visible in HMI	Comment
▼ Input					
MyInput1	Bool	false	☑	☑	
▼ Output					
MyOutput1	Byte	0	☐	☑	
▼ InOut					
▼ Static					

Display of instance-specific values

In instance data blocks, you can apply the already defined values from the interface of the assigned function block or define instance-specific start values. Values that are applied from the function block cannot be edited. You can replace the grayed-out values with instance-specific values. Values that were already changed instance specific are not grayed out.

Meaning of the columns

The following table shows the meaning of the individual columns: You can show or hide the columns as required.

Column	Description
🏷	Symbol you can click to move or copy the tag. You can, for example, drag-and-drop the tag into a program and use it there as operand.
Name	Name of the tags.
Data type	Data type of the tags.
Offset	Relative address of the tags. The column is only visible in data blocks with standard access.
Default value	Default value of the tag in the interface of a higher-level code block or in a PLC data type. The values contained in the "Default value" column can only be changed in the higher-level code block or PLC data type. The values are only displayed in the data block.
Start value	Value that the tag should assume at startup. The default values defined in a code block are used as start values during the creation of the data block. You can then replace these adopted values with instance-specific start values. Specification of an start value is optional. If you do not specify any value, the tag assumes the default value at startup. If a default is not defined either, the default value valid for the data type is used. For example, the value "FALSE" is specified as standard for BOOL.
Monitor value	Current data value in the CPU. This column only appears if an online connection is available and you click "Monitor".
Snapshot	Shows values that were loaded from the device.
Retain	Marks the tag as retentive. The values of retentive tags are retained even after the power supply is switched off.
Visible in HMI	Shows whether the tag is visible by default in the HMI selection list.
Accessible from HMI	Shows whether HMI can access this tag during runtime. The column is only visible on S7-1200-CPUs.
Comment	Comments to document the tags.

See also

Creating data blocks (Page 846)

Basic information on start values (Page 1023)

Creating data blocks

Requirements

The "Program blocks" folder in the project tree is open.

Procedure

To create a data block, follow these steps:

1. Double-click the "Add new block" command.

   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Select the type of the data block. You have the following options available to you:
   - To create a global data block, select the list entry "Global DB".
   - To create an instance data block, select the function block to which you want to assign the instance data block from the list. The list contains only the function blocks that were previously created for the CPU.
   - To create a data block based on a PLC data type, select the PLC data type from the list. The list contains only the PLC data types that were previously created for the CPU.
   - To create a data block based on a system data type, select the system data type from the list. The list contains only those system data types that have already been inserted to program blocks in the CPU.
4. Enter a name for the data block.
5. Enter the properties of the new data block.
   - Select whether you want to assign the block number manually or automatically. Enter a number if you decide to use manual assignment.
   - Select the type of the block access (for S7-1200 only).
6. To enter additional properties of the new data block, click "Additional information".

   An area with further input fields is displayed.
7. Enter all the properties you require.
8. Activate the "Add new and open" check box if the block does not open as soon as it is created.
9. Confirm your entry with "OK".

Result

The new data block is created. You can find the data block in the project tree in the "Program blocks" folder.

See also

Instance data blocks (Page 699)

Global data blocks (DB) (Page 698)

Overview of block properties (Page 854)

Updating data blocks

Introduction

Changes in the interface of a function block or a PLC data type can lead to the corresponding data blocks becoming inconsistent. These inconsistencies are marked in red in the declaration table and at the call point of the block. To remedy these inconsistencies, the data blocks must be updated.

You have three options to update block calls:

- Explicit updating in the declaration table for data blocks.

  The data block is updated. Changes from the interface of the assigned function block and changes to the used PLC data types are applied.
- Explicit updating in the program editor.

  The block calls in the open block will be updated. The associated instance data block is also adjusted in the process.
- Implicit updating during compilation.

  All block calls in the program as well as the used PLC data types and the corresponding instance data blocks are updated.

Explicit Updating in the Declaration Table for Data Blocks

To explicitly update an individual data block, follow these steps:

1. Open the data block.
2. Select "Update interface" in the shortcut menu.

Explicit Updating in the Program Editor

To update all block calls or a specific call within a block, follow these steps:

1. Open the block in the program editor.
2. Right-click on the instruction with the block call.
3. Select the "Update" command in the shortcut menu.
4. The "Interface update" dialog opens. This dialog shows the differences between the block interface in use and the changed interface of the called block.
5. If you want to update the block call, click "OK". To cancel the update, click "Cancel".

Implicit Updating during Compilation

To implicitly update all block calls and uses of PLC data types as well as the instance data blocks during the compiling, follow these steps:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Changing the properties of tags in instance data blocks (Page 1029)

Creating a data structure for global data blocks

Declaring tags of elementary data type

Requirement

A global data block is open.

Note

You cannot change the structure of instance data blocks and of data blocks based on a PLC data type directly, since the structures of these blocks are defined by the respective function block or the PLC data type.

The type of the data block is entered in the block properties.

Procedure

To declare a tag of the elementary data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. In the "Data type" column, click the button for the data type selection.

   A list of the permissible data types is opened.
3. Select the desired data type.
4. Optional: Change the properties of the tags that are displayed in the other columns.
5. Repeat steps 1 to 4 for all tags that are to be declared.

See also

Displaying and editing block properties (Page 858)

Declaring tags of the ARRAY data type (Page 1020)

Declaring tags of STRUCT data type (Page 1021)

Declaring tags of the ARRAY data type

Requirement

A global data block is open.

Procedure

To declare a tag of the ARRAY data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the "Array" data type in the "Data type" column. You will be supported by autocompletion in this step.

   The "Array" dialog opens.
3. In the "Data type" text box, specify the data type of the array elements.
4. In the "ARRAY limits" text box, specify the high and low limit for each dimension.

   Example of a one-dimensional ARRAY:

   ```
   [0..3]
   ```

   Example of a three-dimensional ARRAY:

   ```
   [0..3, 0..15, 0..33]
   ```
5. Confirm your entry.
6. Optional: Change the properties of the tags that are displayed in the other columns.

Entering start values of ARRAY elements

To set default start values for the individual elements of an ARRAY, follow these steps:

1. Click the triangle in front of the ARRAY data type tags.

   The ARRAY opens and the individual ARRAY elements are shown in separate rows.
2. Enter the required value in the "Start value" column.

Declaring tags of STRUCT data type

Requirement

A global data block is open.

Procedure

To declare a tag of the STRUCT data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter "Struct" in the "Data type" column. You will be supported by autocompletion during input.

   An empty, indented row is inserted after the new tag.
3. Insert the first structural element in the first empty row.

   An additional empty row is inserted after the element.
4. Select a data type for the structure element.
5. Optional: Change the properties of the structural element that is displayed in the other columns of the block interface.
6. Repeat the step 4 to 7 for all additional structure elements.

   It is not necessary to end the structure explicitly. The structure ends with the last element that is entered.
7. To insert a new tag after the structure, leave a blank row after the end of the structure and then start the new tag in the second empty row.

Result

The tag of STRUCT data type is created.

Enter start values of structure elements

To set default start values for the individual elements of a structure, follow these steps:

1. Click the triangle in front of the STRUCT data type tags.

   The structure opens and the individual structure elements are shown in separate rows.
2. Enter the required value in the "Start value" column.

See also

STRUCT (Page 768)

Declaring tags based on a PLC data type

Requirements

- A global data block is open.
- A PLC data type is declared in the current CPU.

Procedure

To declare a tag based on a PLC data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the PLC data type in the "Data type" column. You will be supported by autocompletion during input.
3. Optional: Change the properties of the tags that are displayed in the other columns of the table.

Result

The tag is created.

See also

Layout of the block interface (Page 876)

Declaring higher-level tags

Introduction

To access data areas within a declared tag, you can overlay the declared tags with an additional declaration. This provides you with the option of addressing an already declared tag with a different data type. You can, for example, address the individual bits of a tag of WORD data type with an ARRAY of BOOL.

Overlaying tags

To overlay a tag with a new data type, follow these steps:

1. Open a global data block.
2. Select the tag that you want to overlay with a new data type.
3. Click "Add row" in the toolbar.

   A row is inserted after the tag to be overlaid.
4. Enter a tag name in the "Name" column.
5. Enter the "AT" entry in the "Data type" column. You will be supported by Autocomplete in this step.

   The following is added to the entry in the "Name" column.

   ```
   "AT<Name of the higher-level tag>"
   ```
6. Click the data type selection button again and select the data type for the new tag.

   The tag is created. It points to the same data as the higher-level tag, however interprets this data with the new data type.

Removing overlay

To remove the overlay of a tag, follow these steps:

1. Select the overlaid tag that you want to remove.
2. Select the "Delete" command in the shortcut menu.

   The overlay is removed.

See also

Overlaying tags with AT (Page 731)

Define start values

Basic information on start values

Definition

The start value of a tag is a value defined by you which the tag assumes after a CPU startup.

The retentive tags have a special status. Their values take the defined start value only after a "cold restart". After a "warm restart", they retain their values and are not reset to the start value.

Default values and instance-specific start values

The structure of the data blocks can be derived from higher-level elements.

- An instance data block is based, for example, on the interface of a higher-level code block.
- A global data block can be based on a predefined PLC data type.

In this case you can define a default value for each tag in the higher-level element. These default values are used as start values during the creation of the data block. You can then replace these values with instance-specific start values in the data block.

Specification of an start value is optional. If you do not specify any value, the tag assumes the default value at startup. If a default is not defined either, the default value valid for the data type is used. For example, the value "FALSE" is specified as standard for BOOL.

See also

Define start values (Page 1024)

Structure of the declaration table for data blocks (Page 1015)

Declaring local tags in the block interface (Page 881)

Apply monitor values as start values (Page 1035)

Define start values

Define start values

To define the start values for the tags of a data block, follow these steps:

1. Open the data block.

   The "Default value" column shows the default values that were defined for the tags in the interface of a higher-level code block or in a PLC data type.

2. Click the "Expanded mode" button to show all elements of structured data types.
3. Enter the desired start values in the "Start value" column. The value must match the data type of the tag and should not exceed the range of the data type.

   The start values are defined. The tag takes the defined value at startup, provided it was not declared as retentive.

Resetting a tag to the default value

To reset a tag for which you have defined a start value to the default value, follow these steps:

1. Select a modified value in the table.
2. Delete the value.

   The default value is entered. The default value is displayed.

Resetting all tags to the default value

To reset to the default value all tags for which you have defined an start value, follow these steps:

1. In the shortcut menu, select "Reset start values".

See also

Basic information on start values (Page 1023)

Apply monitor values as start values (Page 1035)

Setting retentivity

Retentivity of tags in data blocks

Retentive behavior

To prevent data loss in the event of power failure, you can mark the data as retentive. This data is stored in a retentive memory area. The options for setting the retentivity depend on the type of data block and the type of block access that is set.

See also

Setting retentivity in an instance data block (Page 1025)

Setting retentivity in a global data block (Page 1026)

Setting retentivity in an instance data block

Introduction

In an instance data block, the editability of the retentive behavior depends on the type of access of the higher-level function block:

- Function block with standard access

  You can define the instance data both as retentive or non-retentive. Individual retentivity settings are not possible for individual tags.

- Function block with optimized access

  In the instance data block, you can define the retentivity settings of the tags that are selected in the block interface with "Set in IDB". With these tags also, you cannot individually set the retentive behavior for each tag. The retentivity setting has an impact on all tags that are selected in the block interface with "Set in IDB".

Setting Retentivity for Standard Access

To centrally set the retentivity of all tags in the data block with standard access, follow these steps:

1. Open the instance data block.
2. Select the check box in the "Retain" column of a tag.

   All tags are defined as retentive.
3. To reset the retentivity setting for all tags, clear the check box in the "Retain" column of a tag.

   All tags will be defined as non-retentive.

Setting Retentivity for Optimized Access

To set the retentive behavior of the tags that are selected with "Set in IDB" in data blocks with optimized access, follow these steps:

1. Open the instance data block.
2. Select the check box in the "Retain" column of a tag.

   All tags selected with "Set in IDB" in the data block interface are defined as retentive.
3. To reset the retentivity setting for the tags, clear the check box in the "Retain" column of a tag.

   All tags selected with "Set in IDB" in the data block interface will be defined as non-retentive.

See also

Basics of block access (Page 700)

Retentivity of tags in data blocks (Page 1025)

Setting retentivity in a global data block

Introduction

In a global data block, the editability of the retentive behavior depends on the type of access:

- Global data block with standard access

  You can define the data both as retentive or non-retentive. Individual retentivity settings are not possible for individual tags.
- Global data block with optimized access

  You can individually define the retentivity settings of the tags. For tags with structured data types, retentivity settings are transferred for all tag elements.

Setting Retentivity for Standard Access

To centrally set the retentivity of all tags in the data block with standard access, follow these steps:

1. Open the global data block.
2. Select the check box in the "Retain" column of a tag.

   All tags are defined as retentive.
3. To reset the retentivity setting for all tags, clear the check box in the "Retain" column of a tag.

   All tags will be defined as non-retentive.

Setting Retentivity for Optimized Access

To individually set the retentivity of all tags in data blocks with optimized access, follow these steps:

1. Open the global data block.
2. In the "Retain" column, select the check box for the tags for which you want to set a retentive behavior.

   The selected tag is defined as retentive.
3. To reset the retentivity setting for the tags, clear the check box in the "Retain" column of a tag.

   All tags selected with "Set in IDB" in the data block interface will be defined as non-retentive.

See also

Basics of block access (Page 700)

Retentivity of tags in data blocks (Page 1025)

Editing the properties of tags in data blocks

Properties of the tags in data blocks

Properties

The following table provides an overview of the properties of tags in data blocks:

Group	Property	Description
General	Name	Name of the tags.
	Data type	Data type of the tags.
	Default value	Default value of the tag in the interface of a higher-level code block or in a PLC data type. The values contained in the "Default value" column can only be changed in the higher-level code block or PLC data type. The values are only displayed in the data block.
	Start value	Value that the tag should assume at CPU startup. The default values defined in a code block are used as start values during the creation of the data block. You can then replace these adopted values with instance-specific start values. Specification of an start value is optional. If you do not specify any value, the tag assumes the default value at startup. If a default is not defined either, the default value valid for the data type is used. For example, the value "FALSE" is specified as standard for BOOL.
	Comment	Comment on the tag.
Attributes	Retain	Marks the tag as retentive. The values of retentive tags are retained even after the power supply is switched off. This attribute is only available in the interface of the function block with optimized access.
	Visible	Indicates whether a parameter is visible in CFC.
	Can be set	Indicates whether a parameter is configurable in CFC.
	For test	Indicates whether a parameter is registered for the CFC test mode.
	Interconnectable	Indicates whether a parameter is interconnectable in CFC.

See also

Changing the properties of tags in instance data blocks (Page 1029)

Changing the properties of tags in global data blocks (Page 1030)

Changing the properties of tags in instance data blocks

Instance-specific tag properties

Two options are available for defining the tag properties:

- The tag properties are applied from the interface of the assigned function block.

  Properties that are applied from the function block are displayed grayed out out in the columns of the declaration table. The "Name" and "Data type" properties are always applied.

- You define instance-specific properties.

  You can change some properties instance specific. Changeable values are, for example, "Comment" or "Visible in HMI". Properties that were changed instance specific are not grayed out in the columns of the declaration table. The instance-specific changes are retained, even if the interface of the higher-level function block is changed and the instance data blocks are subsequently updated.

Editing properties in the declaration table

To edit the properties of one or more tags, follow these steps:

1. Open the instance data block.
2. Change the entries in the columns.

Editing properties in the properties window

To edit the properties of an individual tag, follow these steps:

1. Select a tag in the table.
2. Select the "Properties" command in the shortcut menu.

   The properties window opens. It shows the properties of the tag in the "General" and "Attributes" areas.

3. Select the required area in the area navigation.
4. Change the entries in the text boxes.

Reset individual properties to the default value.

To reset individual tag properties to the value that was defined as default in the function block, follow these steps:

1. Select an instance-specific, modified value in the table.
2. Delete the value.

   The instance-specific value will be deleted and the default value from the interface of the function block entered. The default value is displayed grayed out.

See also

Updating data blocks (Page 1018)

Properties of the tags in data blocks (Page 1028)

Changing the properties of tags in global data blocks

Introduction

Two options are available for defining the tag properties:

- The tag properties are applied from the PLC data type.

  Properties that are applied from the PLC data type are shown grayed out in the columns of the declaration table. The "Name" and "Data type" properties are always applied.

- You define specific properties.

  You can change some properties in the global data block. Changeable values are, for example, "Comment" or "Visible in HMI". Properties that were changed are not grayed out in the columns of the declaration table. The changes are retained, even if the PLC data type changes and the global data block is subsequently updated.

Editing properties in the declaration table

To edit the properties of one or more tags, follow these steps:

1. Open the global data block.
2. Change the entries in the columns.

Editing properties in the properties window

To edit the properties of an individual tag, follow these steps:

1. Select a tag in the table.
2. Select the "Properties" command in the shortcut menu.

   The properties window opens. It shows the properties of the tag in the "General" and "Attributes" areas.
3. Select the required area in the area navigation.
4. Change the entries in the text boxes.

Reset individual properties to the default value.

To reset individual tag properties to the value that was defined as default in the PLC data type, follow these steps:

1. Select a modified value in the table.
2. Delete the value.

   The default value from the PLC data type is entered. The default value is displayed grayed out.

See also

Properties of the tags in data blocks (Page 1028)

Editing the declaration table for data blocks

Inserting table rows

Procedure

Proceed as follows to insert a row above the selected row:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button on the toolbar of the table.

Result

A new row is inserted above the selected row.

Inserting table rows

Procedure

Proceed as follows to insert a row below the selected row:

1. Select the row below which you want to insert a new row.
2. Click the "Add row" button on the table toolbar.

Result

A new empty row will be inserted below the selected row.

Deleting tags

Requirements

A global data block is open.

Procedure

To delete a tag, follow these steps:

1. Select the row with the tag to be deleted. You can also select several rows by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last row.
2. Select the "Delete" command in the shortcut menu.

Note

You cannot directly change the structure of instance data blocks and of global data blocks based on a PLC data type, since the structures of these blocks are defined in the higher-level object.

The type of the data block is entered in the block properties.

See also: Displaying and editing block properties (Page 858)

Automatically filling in successive cells

You can load the contents of one or several table cells into the cells below, automatically filling in the successive cells.

If you automatically fill in cells in the "Name" column, a consecutive number will be appended to each name. For example, "Motor" will become "Motor_1".

You can define individual or more cells as well as entire rows as source area.

If less rows exist in the open table than you want to fill, then you will first have to insert additional empty rows.

Requirement

- The table is open.
- Sufficient declaration rows are available.

Procedure

To automatically fill in successive cells, follow these steps:

1. Select the cells to be loaded.
2. Click the "Fill" symbol in the bottom right corner of the cell.

   The mouse pointer is transformed into a crosshair.
3. Keep the mouse button pressed and drag the mouse pointer downwards over the cells that you want to fill in automatically.
4. Release the mouse button.

   The cells are filled in automatically.
5. If entries are already present in the cells that are to be automatically filled in, a dialog appears. In this dialog you can indicate whether you want to overwrite the existing entries or insert new rows for the new tags.

Show and hide table columns

You can show or hide the columns in a table as needed.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. In the shortcut menu, select the "Show/hide columns" command.

   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.

Editing tags with external editors

To edit individual tags in external editors outside the TIA portal, you can export or import these tags using copy & paste. However, you cannot copy structured tags to an editor.

Requirement

The data block and an external editor are opened.

Procedure

To export and re-import individual tags by drag-and-drop operation, follow these steps:

1. Select one or more tags.
2. Select "Copy" in the shortcut menu.
3. Switch to the external editor and paste the copied tags.
4. Edit the tags as required.
5. Copy the tags in the external editor.
6. Switch back to the declaration table.
7. Select "Paste" in the shortcut menu.

Monitoring data values online

Monitoring data values in data blocks online

You can monitor the current data values of the tags in the CPU directly in the declaration table.

Requirement

- An online connection is available.
- The data block has been loaded to the CPU.
- The program execution is active (CPU in "RUN").
- The data block is open.

Procedure

To monitor the data values, proceed as follows:

1. Start monitoring by clicking the "Monitor all" button.

   The additional "Monitor value" column is displayed in the table. This shows the current data values.

   See also: Structure of the declaration table for data blocks (Page 1015)
2. End the monitoring by clicking the "Monitor all" button again.

Apply monitor values as start values

You can apply the monitor values currently assigned to the tags in the CPU as start values in the data block.

Note

The values that are shown in the "Monitor value" column are always copied. During copying, it is not checked whether all values stem from the same cycle.

Requirement

- An online connection to the CPU is available.
- The data block has been loaded to the CPU.
- The program execution is active (CPU in "RUN").
- The global data block is open.

Procedure

To apply the data values as start values, follow these steps:

1. Start monitoring by clicking the "Monitor all" button.

   The "Monitor value" column is displayed in the table. This shows the current data values.
2. On the toolbar, click "Snapshot of monitored values".

   The latest monitored values will be applied in the "Snapshot" column.
3. End the monitoring by clicking the "Monitor all" button again.
4. Select a value in the "Snapshot" column.
5. Select "Copy" in the shortcut menu.
6. Select a value in the "Start value" column.
7. Select "Paste" in the shortcut menu.
8. Compile and reload the block.

Result

The values are applied as start values. The tags apply the new values at the next startup.

See also

Basic information on start values (Page 1023)

Define start values (Page 1024)

Displaying data values loaded from the device

During the loading of a data block from a device, the current tag values are also loaded. You can display these values.

Requirement

A data block was loaded from the device.

Procedure

To display the current values, follow these steps:

1. Open the data block.
2. Click a column header.
3. In the shortcut menu, select the "Show/hide columns" command.

   The selection of available columns is displayed.
4. Select the check box in the "Snapshot" column.

Result

The current values will be applied in the "Snapshot" column.

> **Note**
>
> If you subsequently change the structure of the data block, the display of the current values gets lost. The "Snapshot" column will then be empty.

9.1.4.4 Programming PLC data types

Basics of PLC data types

Description

PLC data types are data structures that you define and that can be used multiple times within the program. The structure of a PLC is made up of several components, each of which can contain different data types. You define the type of components during the declaration of the PLC data type.

PLC data types can be used for the following applications:

- PLC data types can be used as data types for variables in the variable declaration of logic blocks or in data blocks.
- PLC data types can be used as templates for the creation of global data blocks with identical data structures.

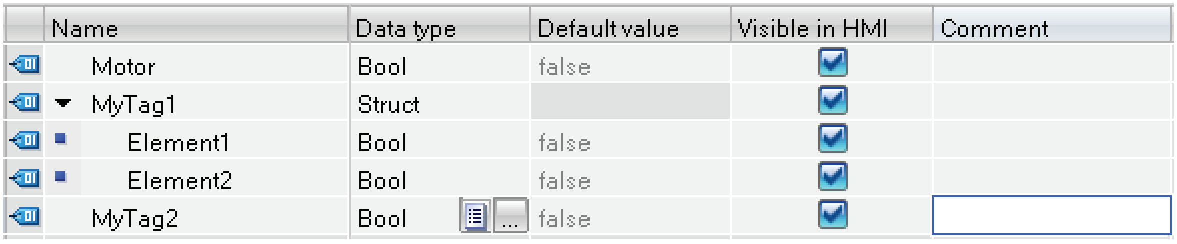
See also

Creating PLC data types (Page 1038)

Structure of the declaration table for PLC data types

Structure of the declaration table for PLC data types

The figure below shows the structure of the declaration table for PLC data types.



	Name	Data type	Default value	Visible in HMI	Comment
◁	Motor	Bool	false	☑	
◁	▾ MyTag1	Struct		☑	
◁	▪ Element1	Bool	false	☑	
◁	▪ Element2	Bool	false	☑	
◁	MyTag2	Bool	false	☑	

Meaning of the columns

The following table shows the meaning of the individual columns: You can show or hide the columns as required.

Column	Description
◁	Symbol you can click to move or copy the tag.
Name	Name of the tags.
Data type	Data type of the tags.
Default value	Value with which you predefine the tag in the declaration of the PLC data type. Specification of the default value is optional. If you do not specify any value the predefined value for the indicated data type is used. For example, the value "false" is predefined for BOOL.
Visible in HMI	Shows whether the tag is visible by default in the HMI selection list.
Accessible from HMI	Shows whether HMI can access this tag during runtime. The column is only visible on S7-1200-CPUs.
Comment	Comments to document the tags.

See also

Creating PLC data types (Page 1038)

Show and hide table columns (Page 1047)

Creating PLC data types

Requirement

The "PLC data types" folder opens in the project tree.

Procedure

To create a PLC data type, proceed as follows:

1. In the "PLC data types" folder, click the "Add new data type" command.

   A new declaration table for creating a PLC data type will be created and opened.
2. Select the PLC data type and select the "Rename" command in the shortcut menu.
3. Enter the name of the PLC data type.

Result

The new PLC data type is created. You can find the PLC data type in the project tree in the "PLC data types" folder.

See also

Structure of the declaration table for PLC data types (Page 1037)

Basics of PLC data types (Page 1036)

Delete PLC data types

Requirement

The PLC data type you want to delete is not open.

Procedure

To delete a PLC data type, follow these steps:

1. In the project tree, open the "PLC data types" folder.
2. Select the PLC data type to be deleted. You can also select several PLC data types by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last data type.
3. Select the "Delete" command in the shortcut menu.

Note

If you delete a PLC data type, the blocks that use the data type will become inconsistent. These inconsistencies are marked in red in the block used. To remedy these inconsistencies, the data blocks have to be updated.

See also:

Updating the block interface (Page 888)

Updating data blocks (Page 1018)

Programming the structure of PLC data types

Declaring tags of elementary data type

Requirement

A PLC data type is open.

Procedure

To declare a tag, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the required data type in the "Data type" column. You will be supported by autocompletion during input.
3. Optional: Change the properties of the tags that are displayed in the other columns.
4. Repeat steps 1 to 3 for all tags that are to be declared.

Declaring tags of the ARRAY data type

Requirement

A PLC data type is open.

Procedure

To declare a tag of the ARRAY data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the "Array" data type in the "Data type" column. You will be supported by autocompletion in this step.

   The "Array" dialog opens.
3. In the "Data type" text box, specify the data type of the array elements.
4. In the "ARRAY limits" text box, specify the high and low limit for each dimension.

   Example of a one-dimensional ARRAY:

   ```
   [0..3]
   ```

   Example of a three-dimensional ARRAY:

   ```
   [0..3, 0..15, 0..33]
   ```
5. Confirm your entry.
6. Optional: Change the properties of the tags that are displayed in the other columns.

Note

You cannot define specific default values for ARRAY elements. You can, however, assign them start values at the usage point in the data block.

See also

Structure of the declaration table for PLC data types (Page 1037)

Declaring tags of STRUCT data type

Requirements

A PLC data type is open.

Procedure

To declare a tag of the STRUCT data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter "Struct" in the "Data type" column. You will be supported by autocompletion during input.

   An empty, indented row is inserted after the new tag.
3. Insert the first structural element in the first empty row.

   An additional empty row is inserted after the element.
4. Select a data type for the structure element.
5. Optional: Change the properties of the structural element that is displayed in the other columns.
6. Repeat steps 3 to 5 for all additional structure elements.

   It is not necessary to end the structure explicitly. The structure ends with the last element that is entered.
7. To insert a new tag after the structure, leave a blank row after the end of the structure and then start the new tag in the second empty row.

Result

The tag of STRUCT data type is created.

See also

STRUCT (Page 768)

Structure of the declaration table for PLC data types (Page 1037)

Declaring tags based on a different PLC data type

Requirements

- A global data block is open.
- A PLC data type is declared in the current CPU.

Procedure

To declare a tag based on a different PLC data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the PLC data type in the "Data type" column. You will be supported by autocompletion during input.

Result

The tag is created.

Note

You define the default values of tags within a PLC data type when the PLC data type is created. You cannot change these values at the point of use of the PLC data type.

See also

Basics of PLC data types (Page 1036)

Structure of the declaration table for PLC data types (Page 1037)

Declaring higher-level tags

Introduction

To access data areas within a declared tag, you can overlay the declared tags with an additional declaration. This provides you with the option of addressing an already declared tag with a different data type. You can, for example, address the individual bits of a tag of WORD data type with an ARRAY of BOOL.

Overlaying tags

To overlay a tag with a new data type, follow these steps:

1. Open the PLC data type.
2. Select the tag that you want to overlay with a new data type.
3. Click "Add row" in the toolbar.

   A row is inserted after the tag to be overlaid.
4. Enter a tag name in the "Name" column.
5. Enter the "AT" entry in the "Data type" column. You will be supported by Autocomplete in this step.

   The following is added to the entry in the "Name" column.

   ```
   "AT<Name of the higher-level tag>"
   ```
6. Click the data type selection button again and select the data type for the new tag.

   The tag is created. It points to the same data as the higher-level tag, however interprets this data with the new data type.

Removing overlay

To remove the overlay of a tag, follow these steps:

1. Select the overlaid tag that you want to remove.
2. Select the "Delete" command in the shortcut menu.
3. The overlay is removed.

See also

Overlaying tags with AT (Page 731)

Editing tag properties in PLC data types

Properties of tags in PLC data types

Properties

The following table gives an overview of tag properties in PLC data types:

Group	Property	Description
General	Name	Name of the tags.
	Data type	Data type of the tags.
	Default value	Default value of the tag in the interface of a higher-level code block or in a PLC data type. The values contained in the "Default value" column can only be changed in the higher-level code block or PLC data type. The values are only displayed in the data block.
	Start value	Not relevant in PLC data types
	Comment	Comment on the tag.
Attributes	Retain	Not relevant in PLC data types
	Visible	Indicates whether a parameter is visible in CFC.
	Can be set	Indicates whether a parameter is configurable in CFC.
	For test	Indicates whether a parameter is registered for the CFC test mode.
	Interconnectable	Indicates whether a parameter is interconnectable in CFC.

See also

Changing the properties of tags in PLC data types (Page 1044)

Basics of PLC data types (Page 1036)

Structure of the declaration table for PLC data types (Page 1037)

Changing the properties of tags in PLC data types

Editing general properties in the declaration table

To edit the general properties of one or more tags, follow these steps:

1. Open the PLC data type.
2. Change the entries in the columns.

Editing detailed properties in the properties window

To edit the detailed properties of an individual tag, follow these steps:

1. Select a tag in the table.
2. Select the "Properties" command in the shortcut menu.

   The inspector window shows the properties of the tag in the "General" and "Attributes" areas.
3. Select the required area in the area navigation.
4. Change the entries in the text boxes.

See also

Updating the block interface (Page 888)

Updating data blocks (Page 1018)

Editing the declaration table for PLC data types

Inserting table rows

Procedure

Proceed as follows to insert a row above the selected row:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button on the toolbar of the table.

Result

A new row is inserted above the selected row.

Inserting table rows

Procedure

Proceed as follows to insert a row below the selected row:

1. Select the row below which you want to insert a new row.
2. Click the "Add row" button on the table toolbar.

Result

A new empty row will be inserted below the selected row.

Deleting tags

Procedure

To delete a tag, follow these steps:

1. Select the row with the tag to be deleted. You can also select several rows by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last row.
2. Select the "Delete" command in the shortcut menu.

See also

Updating the block interface (Page 888)

Updating data blocks (Page 1018)

Automatically filling in successive cells

You can load the contents of one or several table cells into the cells below, automatically filling in the successive cells.

If you automatically fill in cells in the "Name" column, a consecutive number will be appended to each name. For example, "Motor" will become "Motor_1".

You can define individual or more cells as well as entire rows as source area.

If less rows exist in the open table than you want to fill, then you will first have to insert additional empty rows.

Requirement

- The table is open.
- Sufficient declaration rows are available.

Procedure

To automatically fill in successive cells, follow these steps:

1. Select the cells to be loaded.
2. Click the "Fill" symbol in the bottom right corner of the cell.

   The mouse pointer is transformed into a crosshair.
3. Keep the mouse button pressed and drag the mouse pointer downwards over the cells that you want to fill in automatically.
4. Release the mouse button.

   The cells are filled in automatically.
5. If entries are already present in the cells that are to be automatically filled in, a dialog appears. In this dialog you can indicate whether you want to overwrite the existing entries or insert new rows for the new tags.

Show and hide table columns

You can show or hide the columns in a table as needed.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. In the shortcut menu, select the "Show/hide columns" command.

   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.

9.1.4.5 Using external source files

Basics of using external source files

Function

The textual programming languages STL and SCL allow you to enter the program code in any ASCII editor and save it as an external source file. This enables you to perform a range of tasks, for example:

- Declaring tags
- Specify block properties
- Programming blocks
- Specifying system attributes for messages

You can import these source files to your project and use them to generate blocks. You can generate a number of different blocks from one source file.

You also have the option of saving existing blocks as external source files.

See also

Rules for programming external source files (Page 1048)

Saving blocks as external source files (Page 1049)

Inserting external source files (Page 1049)

Opening and editing external source files (Page 1050)

Generating blocks from external source files (Page 1051)

Rules for programming external source files

An external source file basically consists of continuous text. To compile the source into blocks, certain structures and syntax rules must however be adhered to.

Rules

The syntax of the instructions in external source files is very similar to that in the creation of user programs in the program editor with STL or SCL. Note, however, the following additional syntax rules:

- Block call

  When calling a block, transfer the parameters in the defined order in the ASCII editor. If you do not, the comment assignments for these lines may not match.

  Enter the parameters in brackets. The individual parameters are separated by a comma.

- Upper or lower case

  The program editor generally disregards upper or lower case. Jump labels are an exception to this. Character string entries are also case-sensitive ("STRING" data type). Keywords are displayed in upper case. For compilation purposes, however, case is disregarded; you can therefore specify keywords in upper or lower case or a mixture of the two.

- Semicolon

  Mark the end of every instruction and every tag declaration with a semicolon. You can enter several instructions per line.

- Forward slashes

  Begin every comment with two forward slashes (//) and end the comment with the <Enter> key.

See also

Basics of using external source files (Page 1047)

Saving blocks as external source files (Page 1049)

Inserting external source files (Page 1049)

Opening and editing external source files (Page 1050)

Generating blocks from external source files (Page 1051)

Saving blocks as external source files

Procedure

To export a block to an external source file, follow these steps:

1. In the project tree, right-click on the block you want to export to an external source file.
2. Select the "Copy as text" command in the shortcut menu.
3. Open an external text editor.
4. Paste the copied text from the clipboard.
5. Save the file with one of the following file name extensions:
   - ".scl" if you wish to generate a source file for SCL
   - ".stl" if you wish to generate a source file for STL

Result

The block has been saved as an external source file. You can include this source file in a project in the TIA portal and use it to generate other blocks. Please note, however, that you can only use STL source files in S7-300/400 CPUs.

See also

Basics of using external source files (Page 1047)

Rules for programming external source files (Page 1048)

Inserting external source files (Page 1049)

Opening and editing external source files (Page 1050)

Generating blocks from external source files (Page 1051)

Inserting external source files

Requirement

- An external source file is available and complies with the syntax and structure rules.
- The "External source files" folder is open in the project tree.

Procedure

Follow these steps to insert an external source file:

1. Double-click on the "Add new external file" command.

   The "Open" dialog box is opened.
2. Navigate to and select existing external source files.
3. Confirm your selection with "Open".

Result

The new source file will be added to the "External source files" folder.

See also

Basics of using external source files (Page 1047)

Rules for programming external source files (Page 1048)

Saving blocks as external source files (Page 1049)

Opening and editing external source files (Page 1050)

Generating blocks from external source files (Page 1051)

Opening and editing external source files

By linking the files with the file name extensions "stl" and "scl" to an editor you will be able to open and edit external source files with these formats directly.

This means you do not need to insert the external source files again after editing.

Linking files with the file name extensions "stl" and "scl" file types to an editor

Proceed as follows to link files with the file name extensions "stl" and "scl" to an editor:

1. Open Windows Explorer.
2. Right-click on an STL file.
3. Select "Properties" in the shortcut menu.

   The "Properties" dialog box opens.
4. Click "Change" in the "File type" area on the "General" tab.

   The "Open with" dialog box opens.
5. Select the text editor you want to link to the "stl" file type.
6. Confirm your selection with "OK".
7. Close the "Properties" dialog with "OK".
8. Repeat steps 2 to 7 with an SCL file.

Opening and editing an external source file

To open an external source file, follow these steps:

1. Open the "External source files" folder in the project tree.
2. Double-click on the external source file you want to open.

   The external source file will open in the linked editor and can be edited.

See also

Basics of using external source files (Page 1047)

Rules for programming external source files (Page 1048)

Saving blocks as external source files (Page 1049)

Inserting external source files (Page 1049)

Generating blocks from external source files (Page 1051)

Generating blocks from external source files

Requirement

- The "External source files" folder is open in the project tree.
- An external source file is available.

Procedure

To generate blocks from an external source file, follow these steps:

1. Open the external source file from which you wish to generate blocks.
2. Select the "Generate blocks" command in the "Edit" menu.
3. A prompt will appear telling you any existing blocks will be overwritten.
4. Confirm the safety prompt with "Yes".

Result

The external source file blocks will be generated and inserted in the "Program blocks" folder in the project tree. In the event of errors, information about the errors which have occurred will be displayed in the inspector window. This information, however, relates to the external source file and not to the block generated.

See also

Basics of using external source files (Page 1047)

Rules for programming external source files (Page 1048)

Saving blocks as external source files (Page 1049)

Inserting external source files (Page 1049)

Opening and editing external source files (Page 1050)

9.1.5 Comparing project data

9.1.5.1 Basics of comparing project data

Introduction to comparing project data

Function

You can compare the following objects to establish any differences:

- Code blocks with other code blocks
- Data blocks with other data blocks
- PLC tag tables with other PCL tag tables
- PLC tags of a PLC tag table with the PLC tags of another PLC tag table
- PLC data types with other PLC data types
- Tags of a PLC data type with the tags of another PLC data type

Devices and the folder "Program blocks" act as starting points for the comparison. You must start a comparison at one of these starting points. The comparison will cover all underlying objects.

Types and levels of comparison

Two different basic types of comparison can be used:

- Online/offline comparison:

  The objects in the project are compared with the objects of the corresponding device. An online connection to the device is necessary for this comparison. With an Online/Offline comparison you can compare only code and data blocks.
- Offline/offline comparison:

  The objects of two devices either within a project or from various different projects are compared. No online connection is required for this comparison.

Please note that you cannot carry out an unlimited number of comparisons at the same time. The limit is one comparison for each type (online/offline or offline/offline) and for each starting point (device or "Program blocks" folder).

You can choose between the following levels of comparison depending on how in-depth an object comparison you require:

- Comparison editor
- Detailed comparison

When you start a comparison, you will first receive an overview in the comparison editor. For some comparison objects, you can then start a detailed comparison in which the objects compared will be opened side-by-side, each in its own program editor instance. Any differences will be highlighted.

The table below gives an overview of the types and levels of comparison you can apply for each object:

Object	Online/offline		Offline/offline	
	Comparison editor	Detailed comparison	Comparison editor	Detailed comparison
LAD block	X	X	X	X
FBD block	X	X	X	X
STL block	X	X	X	X
LAD block	X	S7-1200 V2 only	X	X
GRAPH block	X	-	X	-
Global data block	X	X	X	X
Instance data block	X	X	X	X
PLC tag table	-	-	X	-
PLC data type	-	-	X	-
Legend: X: available -: not available				

Note

You cannot perform a detailed comparison for know-how protected blocks.

See also

Comparison of code blocks (Page 1054)

Comparison of data blocks (Page 1055)

Comparing PLC tag tables and PCL data types (Page 1055)

Carrying out an online/offline comparison (Page 1056)

Carrying out offline/offline comparisons (Page 1057)

Comparison of code blocks

Introduction

The blocks to be compared in a code block comparison are assigned for comparison on the basis of the following criteria:

- Online/offline comparison: Addresses, e.g. FB100
- Offline/offline comparison: Symbolic names of the blocks

The comparison involves an evaluation of the block time stamps. The results are displayed as an overview in the comparison editor. You can then use actions to define what is to be done about the differences. You can also start detailed comparisons for the individual blocks. The versions of a block compared are opened beside each other and the differences are highlighted.

For the comparison of code blocks, both the block interfaces and the individual networks are compared. Any differing tag names are also determined. All comments and other block attributes are excluded from the comparison.

If the block interface changes, the time stamp of the code block interface will also change. This change means a change in the time stamp of the program code. The first step in comparing block interfaces is therefore a comparison of the program code time stamps. If these time stamps are the same, it is assumed that the interfaces are the same. If the time stamps of the interfaces differ, the next step is to compare the data types of the interfaces, section by section. Multi-instances and PLC data types are included in the comparison. If the data types in the sections are the same, the start values of the tags are compared. All differences are displayed.

When networks are compared, first inserted or deleted networks are detected. Then the other networks are compared. Instructions are the same if the operator and operand are the same. The first difference in each instruction is displayed. However, several differences per network can be displayed.

See also

Introduction to comparing project data (Page 1052)

Comparison of data blocks (Page 1055)

Comparing PLC tag tables and PCL data types (Page 1055)

Carrying out an online/offline comparison (Page 1056)

Carrying out offline/offline comparisons (Page 1057)

Comparison of data blocks

Introduction

The blocks to be compared in a data block comparison are assigned for comparison on the basis of the following criteria:

- Online/offline comparison: Addresses, e.g. DB100
- Offline/offline comparison: Symbolic names of the blocks

The first step in data block comparison is comparing the time stamps of the data block. If these time stamps are the same, it is assumed that the data structures are the same. If the time stamps differ, the structures are then compared until the first difference is found. If the data structures in the sections are the same, the initial and current values of the tags are then compared. All differences are displayed. Any differing tag names are also determined. Comments and PLC data type structures used in the data block are not included in the comparison.

See also

Introduction to comparing project data (Page 1052)

Comparison of code blocks (Page 1054)

Comparing PLC tag tables and PCL data types (Page 1055)

Carrying out an online/offline comparison (Page 1056)

Carrying out offline/offline comparisons (Page 1057)

Comparing PLC tag tables and PCL data types

Introduction

The device PLC tag tables and the device PLC data types will also be shown in the comparison editor if you carry out an offline/offline comparison. The PLC tag tables and the PLC data types will be matched by name and you will receive the following information:

- Status: A symbol shows whether the PLC tag tables / PLC data types are identical or differ.
- Missing PLC tag tables / PLC data types: You can see at a glance whether the PLC tag tables / PLC data types are available in both devices.
- Number of tags: The number of tags each PLC tag table contains is displayed.

You cannot define actions for PLC tag tables and PLC data types.

See also

Introduction to comparing project data (Page 1052)

Comparison of code blocks (Page 1054)

Comparison of data blocks (Page 1055)

Carrying out an online/offline comparison (Page 1056)

Carrying out offline/offline comparisons (Page 1057)

9.1.5.2 Carrying out an online/offline comparison

Requirement

The project tree is open.

Procedure

To perform an online-offline comparison, follow these steps:

1. Select a device in the project tree.
2. Select the "Compare > Offline/online" command in the shortcut menu.
   - If you have already started a comparison of this type for the device selected and would like to replace this comparison with the new comparison, confirm the message which appears by clicking on "Yes". If you click on "No", the new comparison will be cancelled.
   - If you have not yet established an online connection, the "Go online" dialog opens. If the online connection has already been defined, the comparison editor opens.

Result

All objects that exist online and offline are displayed. The symbols in the comparison editor and in the project tree show you the status of the objects. You can now define the actions you require for the objects or start detailed comparisons.

See also

Carrying out offline/offline comparisons (Page 1057)

9.1.5.3 Carrying out offline/offline comparisons

Requirement

The project tree is open.

Comparing the blocks of two different devices in the same project

To compare the blocks of two devices within a project, follow these steps:

1. Select a device or the folder "Program blocks" in the project tree.
2. Select the "Compare > Offline/offline" command in the shortcut menu.

   If you have already started a comparison for this comparison type, a message is displayed. To replace the previous comparison with the new comparison, acknowledge the message shown with "Yes". The "Offline/offline comparison" dialog will open. If you click on "No", the new comparison will be cancelled.
3. Click on the "Device in this project" button.

   The devices available in the project will be listed.
4. Select the device with which you wish to carry out the comparison.
5. Click "OK."

   The comparison editor will open and symbols will show the results of the offline-offline comparison.

Comparing the blocks of two different devices from two different projects

To compare the blocks of two devices from different projects, follow these steps:

1. Select a device or the folder "Program blocks" in the project tree.
2. Select the "Compare > Offline/offline" command in the shortcut menu.

   If you have already started a comparison for this comparison type, a message is displayed. To replace the previous comparison with the new comparison, acknowledge the message shown with "Yes". The "Offline/offline comparison" dialog will open. If you click on "No", the new comparison will be cancelled.
3. Click on the "Device in another project" button.
4. Enter the project with which you wish to carry out the comparison or select it by clicking on the "Browse" button.

   The devices available in the project selected will be listed.
5. Select the device with which you wish to carry out the comparison.
6. Click "OK."

   The comparison editor will open and symbols will show the results of the offline-offline comparison.

Comparing PLC tags

To compare the PLC tags of two PLC tag tables, follow these steps:

1. Select a device in the project tree.
2. Select the "Compare > Offline/offline" command in the shortcut menu.

   If you have already started a comparison for this comparison type, a message is displayed. To replace the previous comparison with the new comparison, acknowledge the message shown with "Yes". The "Offline/offline comparison" dialog will open. If you click on "No", the new comparison will be cancelled.
3. Specify the project with which you wish to carry out the comparison.
4. Select the device with which you wish to carry out the comparison.
5. Click "OK."

   The comparison editor opens and symbols show the results of the offline-offline comparison of the PLC tag table.
6. Double-click on the PLC tag table to compare the individual PLC tags in the table with the PLC tags of the PLC tag table which have the same names as the comparison device.

   The individual tags in the PLC tag table are listed and the existing differences between each tag and the PLC tags in the same PLC tag table in the other device are displayed. For each PLC tag you can set an action to be executed at synchronization.

Comparing PLC data types

To compare the PLC data types of two devices, follow these steps:

1. Select a device in the project tree.
2. Select the "Compare > Offline/offline" command in the shortcut menu.

   If you have already started a comparison for this comparison type, a message is displayed. To replace the previous comparison with the new comparison, acknowledge the message shown with "Yes". The "Offline/offline comparison" dialog will open. If you click on "No", the new comparison will be cancelled.
3. Specify the project with which you wish to carry out the comparison.
4. Select the device with which you wish to carry out the comparison.
5. Click "OK."

   The comparison editor opens and symbols show the results of the offline-offline comparison of the PLC data types for the devices.
6. Double-click on the PLC data type to compare the tags of the PLC data type with the tags of the PLC data type which have the same names as the comparison device.

   The tags of the PLC data types are listed and the existing differences to the tags of the same PLC data type in the other device are displayed. For each tag you can set an action to be executed at synchronization.

See also

Carrying out an online/offline comparison (Page 1056)

9.1.5.4 Using the comparison editor

Overview of the comparison editor

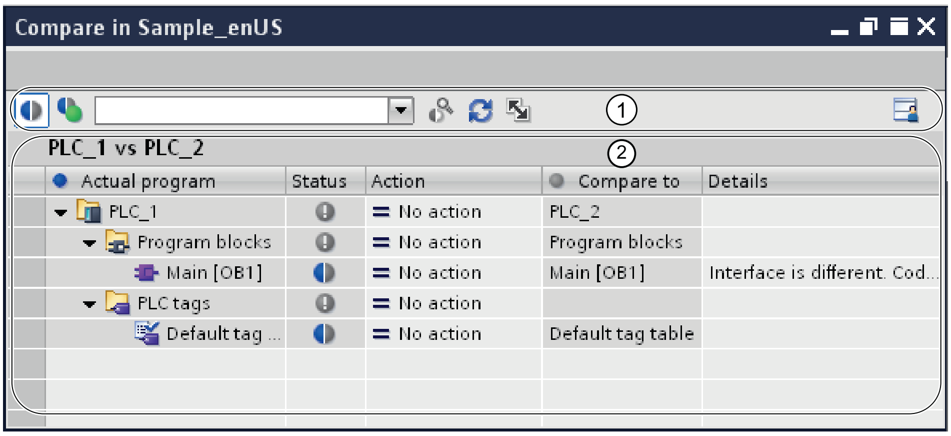
Function

The comparison editor gives an overview of the results of online/offline and offline/offline comparisons in a table. You can also define which actions are to be carried out for non-identical objects compared.

Please note that you cannot carry out an unlimited number of comparisons at the same time. The limit is one comparison for each type (online/offline or offline/offline) and for each starting point (device or "Program block" folder).

Components of the comparison editor

The following figure shows the components of the comparison editor using an offline/offline comparison as an example:



PLC_1 vs PLC_2

● Actual program	Status	Action	● Compare to	Details
PLC_1	!	= No action	PLC_2	
Program blocks	!	= No action	Program blocks	
Main [OB1]	◐	= No action	Main [OB1]	Interface is different. Cod...
PLC tags	!	= No action		
Default tag ...	◐	= No action	Default tag table	

① Comparison editor toolbar

② Tabular comparison overview

Comparison editor toolbar

With the toolbar, you can access the following comparison editor functions:

- Showing and hiding identical objects

  You can hide identical objects to make the comparison easier to follow.

- Scope of the comparison

  You can define which blocks are to be compared.

- Start detailed comparison

  You can start a detailed comparison for objects to show the individual differences. This function is, however, not available for every object.

- Refresh the view

  After you have modified objects, you can update the view of the comparison editor with this function.

- Execute action

  You can synchronize non-identical objects using specific comparison actions.

Comparison editor columns

The following table sets out the meaning of the columns in the comparison editor:

Column	Description
Actual program	Program selected for comparison which is compared with an offline or online version of the program.
Status	Result of the comparison, indicated by symbols
Action	Action for synchronization
Description	Description of the selected action
Compare to	Offline or online version of the program used for comparison.
Path	Path to the comparison object
Details	Details of current differences

Not all columns are shown in the default setting. You can, however, show or hide the columns as required as in all table editors.

Comparison editor symbols

The results of the comparison are indicated by symbols.

The following table shows the comparison results symbols for an online/offline comparison:

Symbol	Description
!	Folder contains objects whose online and offline versions differ
?	Comparison results are not known

Symbol	Description
🟩	Online and offline versions of the object are identical
●	Online and offline versions of the object are different
◐	Object only exists offline
◑	Object only exists online

The following table shows the comparison results symbols for an offline/offline comparison:

Symbol	Description
●	Actual program
○	Version compared
❗	Folder contains objects of which the versions compared differ
❓	Results of the offline/offline comparison are not known
🟩	The versions of the object compared are identical
●	The versions of the object compared differ
◐	Object only exists in the output program
◑	Object only exists in the version compared

The following table shows the symbols for possible actions:

Symbol	Description
⏸	No action
→	Overwrite the object of the compared version with the object from the output program
←	Overwrite the object of the output program with the object from the compared version
⇄	Different actions for the comparison objects in the folder

See also

Carrying out an online/offline comparison (Page 1056)

Carrying out offline/offline comparisons (Page 1057)

Filtering the comparison editor view (Page 1062)

Updating the comparison results (Page 1063)

Filtering the comparison editor view

You can improve the clarity of the comparison editor using the following filters:

- Hiding identical comparison objects

  You can hide comparison objects which have identical online/offline or offline/offline versions. Any such comparison objects you have hidden can also be shown again at any time.

- Block type shown

  You can define the blocks whose comparison results are to be shown.

Requirement

The comparison editor is open.

Hiding identical comparison objects

To hide identical objects, follow these steps:

1. Click on the "Show only objects with differences" button in the toolbar.

   Only the elements that differ online and offline are displayed.

Showing identical comparison objects

To show identical objects again, follow these steps:

1. Click on the "Show identical and different objects" button in the toolbar.

   All elements will be displayed.

Selecting block type to be shown

To select the block type whose comparison results are to be shown, follow these steps:

1. Click on the arrow button in the drop-down list in the toolbar.
2. Select the block type you want.

See also

Carrying out an online/offline comparison (Page 1056)

Carrying out offline/offline comparisons (Page 1057)

Overview of the comparison editor (Page 1059)

Updating the comparison results (Page 1063)

Updating the comparison results

As soon as you change an object, the comparison results are no longer valid and must be updated.

Note

For online/offline comparisons, you should note that changes in the device may result in the system automatically updating the comparison editor if objects in the comparison are affected by the change. This can have the following results:

- Some of the actions you have defined may become invalid, for example if the device no longer contains the object in question. Objects with such invalid actions will be highlighted so you can define new, valid actions.
- The selection you made before the automatic update may also be cancelled.

Requirement

The comparison editor is open.

Procedure

To update the comparison results, follow these steps:

1. Click the "Refresh view" button in the toolbar.

   The comparison results are updated.

   Note

   Please note that the "Refresh view" button will not be available while the comparison editor is loading or synchronizing content.

See also

Carrying out an online/offline comparison (Page 1056)

Carrying out offline/offline comparisons (Page 1057)

Overview of the comparison editor (Page 1059)

Filtering the comparison editor view (Page 1062)

Synchronizing non-identical objects

Specifying actions

If you have performed a comparison, you can specify the actions to be performed for non-identical objects in the comparison editor. You cannot select any actions for identical objects.

Note

Please note the following CPU-specific aspects when defining actions:

- S7-300/400:
  - You can define actions for the "Program blocks" folder, for folders you have created and for individual blocks.
  - Neither SCL nor GRAPH blocks can be loaded from the device to the offline project.
- S7-1200:
  - You can only define actions at a CPU level. This means that the action you define for the CPU will be performed on all underlying objects.
  - SCL blocks cannot be loaded from the device to the offline project.

Requirement

The comparison editor is open.

Procedure

To select an action for a non-identical object, follow these steps:

1. Click on the arrow button in the drop-down list in the "Action" column.
2. Select the action you want.

   The action set will be carried out for the object in question the next time synchronization is performed.

   If you have accidentally changed the action you had selected, you can undo the change before the next synchronization. The shortcut menu is only available for folders if the same actions have been set for all objects in those folders.
3. To restore the previous action selected, right-click on the object or folder.
4. Select the "Restore previous selection" command in the shortcut menu.

See also

Overview of the comparison editor (Page 1059)

Filtering the comparison editor view (Page 1062)

Updating the comparison results (Page 1063)

Synchronizing objects (Page 1065)

Synchronizing objects

Synchronization executes the actions you have specified for non-identical objects.

Requirement

- The comparison editor is open.
- The desired actions have been selected.

Procedure

To synchronize objects, follow these steps:

1. Click the "Execute actions" button in the toolbar.

Result

The actions you specified for the objects are performed.

See also

Overview of the comparison editor (Page 1059)

Filtering the comparison editor view (Page 1062)

Updating the comparison results (Page 1063)

Specifying actions (Page 1064)

9.1.5.5 Performing detailed block comparisons

Starting a detailed comparison

You can start a detailed comparison for blocks. The versions of a block compared are opened beside each other and the differences highlighted.

Note

Please note the following:

- You cannot carry out detailed comparisons of blocks created in the GRAPH programming language.
- You can carry out detailed comparisons for S7-300/400 and S7-1200 V2 for blocks created in the SCL programming language.

Starting detailed comparisons using the comparison editor

To start a detailed comparison for a block using the comparison editor, follow these steps:

1. In the comparison editor, right-click on the block for which you want to perform a detailed comparison.
2. Click on the "Show details on selected object" button in the toolbar.

Starting detailed comparisons in the program editor

To start a detailed comparison for a block in the program editor, follow these steps:

1. Open the block for which you wish to carry out a detailed comparison.
2. Click the "Detailed comparison" button in the toolbar.

Result

One instance of the program editor will be opened for each version of the block compared and the two instances will be displayed side by side. Any differences will be highlighted in color in each version.

See also

Carrying out offline/offline comparisons (Page 1057)

Carrying out an online/offline comparison (Page 1056)

Representation of the detailed comparison (Page 1067)

Navigating in the detailed comparison (Page 1069)

Changing blocks during detailed comparison (Page 1070)

Updating comparison results (Page 1071)

Representation of the detailed comparison

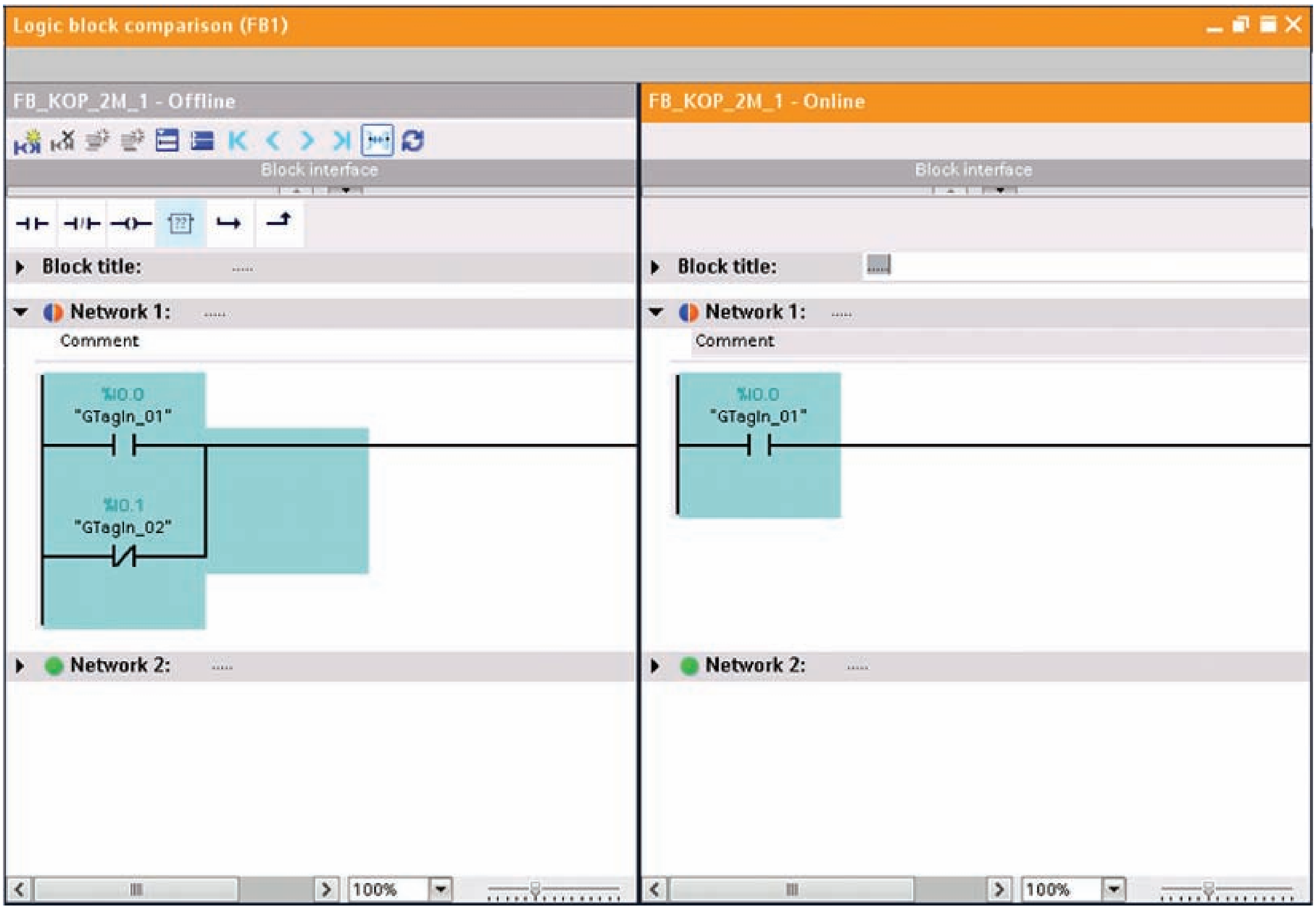
Identification of the differences

The detailed comparison allows you to identify the exact places where versions of a block differ. The following color coding allows you to find these places as quickly as possible:

- The lines where there are differences are highlighted in gray.
- Differing operands and instructions are highlighted in green.
- If the number of networks differs, pseudo-networks are added to allow the display of identical networks to be synchronized. These pseudo-networks are highlighted in grey and contain the text "No corresponding network found" in the title bar of the network. Pseudo-networks cannot be edited.
- If the sequence of the networks is incorrect, pseudo networks will be inserted at the corresponding locations. These pseudo networks are highlighted in gray and contain the text "The networks are not synchronized" in the title bar of the network. The pseudo network also features a "Go to network <No>" link, which you can use to navigate to the corresponding network.

Example

The following figure shows an example of the detailed comparison for the LAD programming language:



Reducing the number of differences displayed

For the sake of clarity, not all the differences are highlighted but rather the first difference of an operation in each case. For example, if all the inputs in a box with multiple inputs are different in the offline and online versions of the block, only the first input is highlighted as a difference. You can resolve this difference and update the comparison list. The next input will then be highlighted as a difference.

The number of differences highlighted within a network therefore depends on the number of instructions.

See also

Carrying out an online/offline comparison (Page 1056)

Carrying out offline/offline comparisons (Page 1057)

Starting a detailed comparison (Page 1065)

Navigating in the detailed comparison (Page 1069)

Changing blocks during detailed comparison (Page 1070)

Updating comparison results (Page 1071)

Navigating in the detailed comparison

Requirement

You have run a detailed comparison.

Navigate to the differences

To navigate to a difference between the two blocks, follow these steps:

1. Open the list of results for the detailed comparison under "Info > Comparison result" in the Inspector window.
2. Double-click a difference.

   The difference is selected in both editors.

Or:

1. Click one of the following navigation buttons on the toolbar:
   – Position on first difference

     Navigates to the first difference in the block, and displays the difference in both editors.

   – Position on previous difference

     Navigates to the previous difference starting from the current position, and displays the difference in both editors.

   – Position on next difference

     Navigates to the next difference starting from the current position, and displays the difference in both editors.

   – Position on last difference

     Navigates to the last difference in the block, and displays the difference in both editors.

Switching off/on the synchronization of the vertical scrolling between the editors

The scrolling for both editors is synchronized to ensure that the corresponding networks are visible parallel to each other during vertical scrolling. You can switch this mode off and on. To do this, follow these steps:

1. To switch off synchronized scrolling, click the "Synchronize scrolling between editors" button in the toolbar.
2. To switch on synchronized scrolling again, click the "Synchronize scrolling between editors" button one more time in the toolbar.

See also

Carrying out an online/offline comparison (Page 1056)

Carrying out offline/offline comparisons (Page 1057)

Starting a detailed comparison (Page 1065)

Representation of the detailed comparison (Page 1067)

Changing blocks during detailed comparison (Page 1070)

Updating comparison results (Page 1071)

Changing blocks during detailed comparison

Changing offline blocks

You can change offline blocks at any time.

Changing online blocks

You cannot change online blocks.

See also

Carrying out an online/offline comparison (Page 1056)

Carrying out offline/offline comparisons (Page 1057)

Starting a detailed comparison (Page 1065)

Representation of the detailed comparison (Page 1067)

Navigating in the detailed comparison (Page 1069)

Updating comparison results (Page 1071)

Updating comparison results

As soon as you change an object, the comparison results are no longer valid and must be updated.

Requirement

You have run a detailed comparison.

Procedure

To update the comparison results, follow these steps:

1. Click "Update the comparison result" in the toolbar.

See also

Carrying out an online/offline comparison (Page 1056)

Carrying out offline/offline comparisons (Page 1057)

Starting a detailed comparison (Page 1065)

Representation of the detailed comparison (Page 1067)

Navigating in the detailed comparison (Page 1069)

Changing blocks during detailed comparison (Page 1070)

9.1.6 Compiling and downloading blocks

9.1.6.1 Compiling blocks

Basic information on compiling blocks

Introduction

A user program must first be compiled before the CPU can execute it. You need to recompile your program each time you make a change.

The following procedures take place during compilation:

- The user program is checked for syntax errors.
- All the block calls within the compiled blocks are checked. In case of changes to the interface of called blocks, errors will be shown in the "Compilation" tab of the information window. You have to correct these errors first.
- Finally, the user program is compiled into a code that can be read by the CPU.

Compilation methods

You can start compilation in the following windows or editors:

- Compiling blocks in the project tree

  Used to compile individual blocks or for the simultaneous compilation of all blocks in the "Program blocks" folder.

- Compiling blocks in the program editor

  This is intended for compilation of a single open block.

- Compiling blocks in the call or dependency structure

  Used to compile individual blocks.

  See also: Call structure (Page 1103), Dependency structure (Page 1111)

Compilation options

If you are compiling blocks in project tree, you have further options:

- Software

  Only the changed blocks are compiled.

- Software (compile all blocks)

  All blocks are compiled. This is recommended for the first compilation and after major revisions.

Consistency check

Changing the interfaces of blocks called or PLC data types used can result in inconsistencies between calling blocks and called blocks or between the PLC data types and the global data blocks which use these PLC data types.

To avoid such inconsistencies in the user program, the system performs an automatic consistency check before each compilation process. The time stamps are compared and compilation is then either carried out or cancelled depending on the results of the comparison.

- The calling block can only be compiled if the time stamps of the interfaces of the called blocks are older than those of the calling block.
- A global data block based on a PLC data type can only be compiled correctly if the time stamp of the global data block is newer than the time stamp of the PLC data type used.
- The instance data block can only be compiled correctly if the interface time stamps for the interface of the instance data block are identical to those of the assigned function block.

If the compilation process is cancelled, a message will be displayed in the inspector window. You must then update the block calls in the relevant blocks and the PLC data types in the global data blocks before re-starting the compilation process. The consistency check also finds know-how protected blocks which cannot be compiled. The corresponding messages will also be shown in the inspector window.

If you start loading immediately instead of first compiling, the blocks selected will be automatically compiled and the block call and global data blocks implicitly updated. Please note the following differences between the CPU families:

- S7-1200: All blocks affected are loaded to ensure no inconsistencies can arise.
- S7-300/400: Only the block selected is loaded.

See also

Compiling blocks in the project tree (Page 1073)

Compiling blocks in the program editor (Page 1074)

Correcting compilation errors (Page 1075)

Block time stamps (Page 856)

Updating block calls in LAD (Page 918)

Updating block calls in FBD (Page 956)

Compiling blocks in the project tree

You can compile one block, multiple blocks or all of the blocks in the project tree.

Requirements

The project tree is open.

Compiling one or more blocks in the project tree

To compile multiple blocks in the project tree, follow these steps:

1. Open the "Program blocks" folder in project tree.
2. Select the blocks you want to compile.
3. Select the "Compile > Software" command in the shortcut menu.

Compiling all blocks in the project tree

To compile all blocks in the "Program blocks" folder in project tree, follow these steps:

1. Select the "Program blocks" folder in the project tree.
2. You can select one of two different options for the compilation:
   - If you want to compile only the changes since the last compilation, select the "Compile > Software" command in the shortcut menu.
   - If you want to compile all blocks completely, select the "Compile > Software (compile all blocks)" command in the shortcut menu.

Result

The code for the blocks will be generated if the consistency check has been successful. Instance data blocks generated by the system which are no longer needed will be deleted.

The message under "Info > Compilation" in the inspector window reports whether the compilation was successful.

See also

Basic information on compiling blocks (Page 1071)

Compiling blocks in the program editor (Page 1074)

Correcting compilation errors (Page 1075)

Finding syntax errors in the program (Page 1008)

Compiling blocks in the program editor

Requirements

The block to be compiled is open.

Procedure

To compile a block in the program editor, follow these steps:

1. Right-click the white area underneath a network in the instruction window of the program editor.
2. Select the "Compile" command in the shortcut menu.

Result

The code for the block is generated. Instance data blocks generated by the system which are no longer needed will be deleted.

The message under "Info > Compilation" in the inspector window reports whether the compilation was successful.

See also

Basic information on compiling blocks (Page 1071)

Compiling blocks in the project tree (Page 1073)

Correcting compilation errors (Page 1075)

Correcting compilation errors

In the Inspector window in "Info > Compile", you can see whether any compilation was successful or whether errors were detected in the program. If errors occur, you will need to correct them and then start the compilation again.

Procedure

To correct errors following compilation, follow these steps:

1. Open the error list in the Inspector window with "Info > Compile".
2. If there is one, click on the blue question mark next to the error text for information on remedying errors.
3. Double-click the error you want to correct.

   The corresponding error is highlighted.
4. Correct the error.
5. Restart compilation.

See also

Basic information on compiling blocks (Page 1071)

Compiling blocks in the program editor (Page 1074)

Compiling blocks in the project tree (Page 1073)

9.1.6.2 Downloading blocks

Introduction to downloading blocks

Downloading blocks to device

So that the CPU can execute the user program, the program must first be compiled and then downloaded to the device. The following options are available for downloading:

- Downloading blocks to the program editor

  You can load a single open block in the program editor.
- Downloading blocks to the project tree

  You can download several or all of the blocks in the block folder via the project tree.
- Downloading blocks to an accessible device

  You can download blocks to an accessible device by dragging them.

Note

To avoid inconsistencies between calling and called blocks, all blocks affected are compiled and loaded after each global change, such as a change in the block interface.

Note

S7-1200 Version 1.0

Please note the following:

- If you download a know-how-protected block to a device, no restore information is loaded along with it. This means that you cannot reopen a know-how-protected block if you upload it from the device.
- If you download an element of your project to the CPU, for example a program block, a data block or the hardware configuration, the CPU runs a cold restart the next time it changes to RUN mode. Apart from deleting the inputs, initializing the outputs and deleting the non-retentive memory, cold restart also deletes the retentive memory areas. All subsequent changes from STOP to RUN are warm restarts in which the retentive memory is not deleted.

Uploading blocks from device

You can upload blocks from an accessible device to your project. This is necessary, for example, if you want to edit blocks that only exist in the device. All available blocks (organization blocks, function blocks, functions and data blocks) and the global PLC tags are uploaded to the project.

Uploading blocks from or downloading blocks to a memory card

Memory cards are plug-in cards used, for example, to expand the memory of a device. Only Siemens SD cards can be used for devices of the S7-1200 series.

To use a memory card as replacement of the load memory, you must download the user program or individual blocks to a memory card. You can just as well upload blocks from a memory card back into the project.

Note

Note the following when uploading to or downloading from a memory card:

- If the CPU contains no previous program and you insert an empty memory card in the CPU the program will be loaded from the PG/PC to the memory card and not to the CPU.
- If you insert an empty memory card prior to the startup of the CPU, the program that is on the CPU will be transferred automatically to the memory card. The program on the CPU will then be deleted.
- If you insert a memory card with a program in the CPU prior to the startup of the CPU and the CPU already contains a program, the program on the memory card will be executed and not the program on the CPU. The program on the CPU will be deleted.

Downloading blocks to device

Downloading blocks from program editor to device

Requirement

The block to be downloaded is open.

Procedure

To download a block from the program editor to the device, follow these steps:

1. Right-click the white area underneath a network in the instruction window of the program editor.
2. Select the "Download to device" command in the shortcut menu.
   - If you have not already established an online connection, the "Extended download to device" dialog will open. Set all the necessary parameters for the connection and click on "Load". You can also open the "Extended download to device" dialog via the "Online" menu.

     See also: Going online and going offline
   - If you have already defined an online connection, the project data will if necessary be compiled and the "Load preview" dialog will open. This dialog displays messages and proposes actions necessary for loading.

3. Check the messages and, where necessary, enable the actions in the "Action" column.

NOTICE
Performing the proposed actions during system operation can cause serious damage to property or injury to persons if there are functional faults or program errors! Make sure before starting the actions that they will not pose a hazard.

   As soon as downloading becomes possible, the "Load" button is enabled.

4. Click on "Load".

   The block is downloaded and the "Load results" dialog opens. This dialog shows you the status and the actions after downloading.

5. If you want to start the modules again directly after downloading, select the "Start all" check box.
6. To close the "Load results" dialog box, click "Finish".

Result

The code for the block will be downloaded to the device. If the changes affect additional blocks, these will be compiled and also downloaded to the device. Blocks that only exist in the device online will be deleted. Inconsistencies between the blocks in the user program are avoided by downloading all blocks affected and deleting the unnecessary blocks in the device.

The messages under "Info > General" in the Inspector window show whether the downloading process was successful.

See also

Downloading blocks from the project tree to the device (Page 1078)

Downloading blocks to an accessible device (Page 1081)

Downloading blocks from the project tree to the device

In the project tree you can download one block, multiple blocks or all blocks to a device.

Downloading one or more blocks from the project tree to the device

To download one block or multiple blocks to the device from the project tree, follow these steps:

1. Open the "Program blocks" folder in project tree.
2. Select the blocks you want to download.
3. Select the "Download to device > Software" command in the shortcut menu.
   – If you have not already established an online connection, the "Extended download to device" dialog will open. Set all the necessary parameters for the connection and click on "Load". You can also open the "Extended download to device" dialog via the "Online" menu.

     See also: Going online and going offline

   – If you have already defined an online connection, the project data will if necessary be compiled and the "Load preview" dialog will open. This dialog displays messages and proposes actions necessary for loading.
4. Check the messages and, where necessary, enable the actions in the "Action" column.

NOTICE
Performing the proposed actions during system operation can cause serious damage to property or injury to persons if there are functional faults or program errors! Make sure before starting the actions that they will not pose a hazard.

   As soon as downloading becomes possible, the "Load" button is enabled.
5. Click on "Load".

   The blocks will be downloaded and the "Load results" dialog will open. This dialog shows you the status and the actions after downloading.
6. If you want to start the modules again directly after downloading, select the "Start all" check box.
7. To close the "Load results" dialog box, click "Finish".

Downloading all blocks from the project tree to the device

To download all blocks in the "Program blocks" folder to the device from the project tree, follow these steps:

1. Select the "Program blocks" folder in the project tree.
2. Select the "Download to device" submenu in the shortcut menu.

3. If you only want to download the changes since the last download, select the "Software" option. If you want to download all blocks in their entirety, select the "Software (all blocks)" option.

   – If you have not already established an online connection, the "Extended download to device" dialog will open. Set all the necessary parameters for the connection and click on "Load". You can also open the "Extended download to device" dialog via the "Online" menu.

     See also: Going online and going offline

   – If you have already defined an online connection, the project data will if necessary be compiled and the "Load preview" dialog will open. This dialog displays messages and proposes actions necessary for loading.

4. Check the messages and, where necessary, enable the actions in the "Action" column.

NOTICE
Performing the proposed actions during system operation can cause serious damage to property or injury to persons if there are functional faults or program errors! Make sure before starting the actions that they will not pose a hazard.

   As soon as downloading becomes possible, the "Load" button is enabled.

5. Click on "Load".

   The blocks will be downloaded and the "Load results" dialog will open. This dialog shows you the status and the actions after downloading.

6. If you want to start the modules again directly after downloading, select the "Start all" check box.

7. To close the "Load results" dialog box, click "Finish".

Result

The code for the blocks is downloaded to the device. If the changes affect additional blocks, these will be compiled and also downloaded to the device. Blocks that only exist in the device online will be deleted. Inconsistencies between the blocks in the user program are avoided by downloading all blocks affected and deleting the unnecessary blocks in the device.

The messages under "Info > General" in the Inspector window show whether the downloading process was successful.

See also

Downloading blocks from program editor to device (Page 1077)

Downloading blocks to an accessible device (Page 1081)

Downloading blocks to an accessible device

Requirement

The accessible devices are displayed.

See also: Displaying accessible devices (Page 3071)

Procedure

To download blocks to an accessible device, follow these steps:

1. Open the "Program blocks" folder of the PLC in the project tree.
2. Select the blocks you want to download to the accessible devices.
3. In the project tree, drag the blocks to the "Program blocks" folder of the accessible device.

   The "Load preview" dialog opens. This dialog displays messages and proposes actions necessary for loading.
4. Check the messages and, where necessary, enable the actions in the "Action" column.

NOTICE
Performing the proposed actions during system operation can cause serious damage to property or injury to persons if there are functional faults or program errors! Make sure before starting the actions that they will not pose a hazard.

   As soon as downloading becomes possible, the "Load" button is enabled.
5. Click the "Load" button.

   The load is executed. The "Load results" dialog will then open. In this dialog, you can check whether or not loading was successful and take any further action that may be necessary.
6. If you want to start the modules again directly after downloading, select the "Start all" check box.
7. Click "Finish".

Result

The blocks selected will be downloaded to the accessible device. If the changes affect additional blocks, these will also be downloaded to the accessible node. Blocks that only exist in the device online will be deleted. Inconsistencies between the blocks in the user program are avoided by downloading all blocks affected and deleting the unnecessary blocks in the device.

The messages under "Info > General" in the Inspector window show whether the downloading process was successful.

See also

Downloading blocks from program editor to device (Page 1077)

Downloading blocks from the project tree to the device (Page 1078)

Uploading blocks from device

Uploading blocks from device

Requirement

The online and offline versions of a block differ or a block only exists online.

Procedure

To upload blocks from a device, follow these steps:

1. Establish an online connection with the device from which you want to download the blocks.

   See also: Establishing and terminating an online connection
2. Select the "Program blocks" folder.
3. In the "Online" menu, select the "Upload from device" command.

   The "Upload preview" dialog box opens. This dialog displays messages and proposes actions necessary for loading.
4. Check the messages and, where necessary, enable the actions in the "Action" column.

   As soon as uploading becomes possible, the "Upload from device" button is enabled.
5. Click the "Upload from device" button.

   The load is executed.

Result

The blocks will be uploaded from the device to the project. You can edit them as normal, recompile them and download them to the device again.

See also

Uploading blocks from an accessible device (Page 1083)

Uploading blocks from an accessible device

Requirement

- The accessible devices are displayed.

  See also: Displaying accessible devices (Page 3071)
- The project includes a device.

Procedure

To upload blocks from an accessible device to your project, follow these steps:

1. In the project tree, drag the accessible device "Program blocks" folder to the "Program blocks" folder of the device in the project.

   The "Upload preview" dialog box will open. This dialog displays messages and proposes actions necessary for loading. If the device folder in the project already contains data, a message will inform you that this data will be replaced.
2. Check the messages and, where necessary, enable the actions in the "Action" column.
3. The "Upload from device" button will be enabled as soon as uploading becomes possible.
4. Click on the "Upload from device" button.

Result

The blocks will be uploaded from the accessible devices to the project. You can edit them as normal and recompile and load them.

See also

Uploading blocks from device (Page 1082)

Uploading blocks from or downloading blocks to a memory card

Downloading blocks to a memory card

Requirement

- The memory card is not marked as program card.
- The "Program blocks" folder of the memory card is open.

Procedure

To download blocks to a memory card, follow these steps:

1. Open the device "Program blocks" folder in the project tree.
2. Select the blocks you want to download to the memory card.
3. Drag the blocks in project tree to the "Program blocks" folder of the memory card.

   The "Load preview" dialog opens. This dialog displays messages and proposes actions necessary for loading.
4. Check the messages and, where necessary, enable the actions in the "Action" column.
5. As soon as downloading becomes possible, the "Load" button is enabled.
6. Click the "Load" button.

   The load is executed. The "Load results" dialog will then open. In this dialog, you can check whether or not loading was successful and take any further action that may be necessary.
7. Click "Finish".

Result

The block is downloaded to the memory card If the changes affect additional blocks, these will also be downloaded to the memory card. Blocks that exist only on the memory card are deleted. Inconsistencies between the blocks in the user program are avoided by downloading all affected blocks and the deleting of the non-required blocks on the memory card.

The messages under "Info > General" in the Inspector window show whether the downloading process was successful.

See also

Uploading blocks from a memory card (Page 1085)

Uploading blocks from a memory card

Requirement

The "Program blocks" folder of the memory card is open.

Procedure

To upload blocks from a memory card to your project, follow these steps:

1. Select the blocks you want to upload in the "Program blocks" folder of the memory card in project tree.
2. Drag the blocks to the device "Program blocks" folder.

   The "Upload preview" dialog box will open. This dialog displays messages and proposes actions necessary for loading.
3. Check the messages and, where necessary, enable the actions in the "Action" column.

   The "Upload from device" button will be enabled as soon as uploading becomes possible.
4. Click on the "Upload from device" button.

See also

Downloading blocks to a memory card (Page 1083)

9.1.7 Protecting blocks

9.1.7.1 Protecting blocks

Introduction

You can use a password to protect one or more blocks of the OB, FB, FC type and global data blocks from unauthorized access.

You can not manually protect instance data blocks; they depend on the know-how protection of the assigned FB. This means that when you create an instance data block for a know-how protected FB, the instance data block also receives this know-how protection. This is independent of whether you explicitly create the instance data block or if it is created by a block call.

If a block is know-how protected, only the following data is readable without the correct password:

- Interface parameters Input, Output, InOut, Return, Static, Temp
- Block title
- Block comment

- Block properties
- Global tags without information on the point of use

The following actions can be performed with a know-how protected block:

- Copying and deleting
- Calling in a program
- Online/offline comparison
- Compiling and loading
- Printing

The code of the block, on the other hand, is protected from unauthorized reading and modification. For S7-1200 CPUs, you can also set up copy protection which binds execution of the block to the CPU or SD card with the defined serial number.

Note

Please note the following:

- S7-1200 Version 1.0: If you download a know-how-protected block to a device, no restore information is loaded along with it. This means that you cannot open a know-how-protected block again even with the correct password if you upload it from the device.
- Only the non-protected data is compared in offline-online comparison of know-how protected blocks.
- You will no longer be able to access the block if you do not have the password.
- If you add a know-how-protected block to a library, the master copy created will also be know-how protected. You therefore require the correct password for the know-how protected block when using copies.

See also

Setting up and removing block copy protection (Page 1087)

Setting up block know-how protection (Page 1088)

Opening know-how protected blocks (Page 1089)

Printing know-how protected blocks (Page 1090)

Removing block know-how protection (Page 1092)

Changing a password (Page 1091)

9.1.7.2 Setting up and removing block copy protection

For S7-1200 CPUs, you can set up copy protection which binds execution of the block to a specific CPU or specific SD card. The block can then only be executed if it is in the device with the set serial number.

It is important that you also know-how-protect any block for which you have set up copy protection. If you do not, anyone can reset the copy protection. You must, however, set up copy protection first as the copy protection settings are read-only if the block is already know-how-protected.

Note

You can only remove the copy protection of a block if the block is not know-how protected. This means that you cannot remove the copy protection of a block with know-how protection that you load from a device, because you cannot undo the know-how protection.

Requirement

The block is not know-how protected.

Setting up copy protection

To set up copy protection for a block, follow these steps:

1. Open the block you wish to copy-protect.
2. Open the "Properties" tab in the inspector window.
3. Select "Protection" in the area navigation in the inspector window.
4. Select either "Bind to serial number of the CPU" or "Bind to serial number of the memory card" from the drop-down list in the "Copy protection" area.
5. Enter either the serial number of the CPU or memory card or enable the option "Serial number inserted when downloading to a device or memory card" if the serial number is to be inserted automatically when you load.
6. You can now set up the know-how protection for the block in the "Know-how protection" area.

Removing copy protection

To remove copy protection, follow these steps:

1. Remove the know-how protection for the block whose copy protection you wish to remove.
2. Open the block.
3. Open the "Properties" tab in the inspector window.
4. Select "Protection" in the area navigation in the inspector window.
5. Select "No binding" in the drop-down list in the "Copy protection" area.

See also

Protecting blocks (Page 1085)

Setting up block know-how protection (Page 1088)

Opening know-how protected blocks (Page 1089)

Printing know-how protected blocks (Page 1090)

Removing block know-how protection (Page 1092)

Changing a password (Page 1091)

9.1.7.3 Setting up block know-how protection

You can set up know-how protection for blocks in the devices in your project.

Procedure

To set up block know-how protection, follow these steps:

1. Select the blocks with no know-how protection which you want to protect.
2. Select the command "Know-how protection" in the "Edit" menu.

   The "Know-how protection" dialog will open.
3. Click "Define".

   The "Define password" dialog box opens.
4. Enter a password in the "New" field.
5. Enter the same password in the "Confirm" field.
6. Confirm your entries with "OK".
7. Close the "Know-how protection" dialog by clicking on "OK".

Result

The blocks selected will be know-how-protected. Know-how protected blocks are marked with a lock in the project tree. The password entered is valid for all blocks selected.

See also

Protecting blocks (Page 1085)

Setting up and removing block copy protection (Page 1087)

Opening know-how protected blocks (Page 1089)

Printing know-how protected blocks (Page 1090)

Removing block know-how protection (Page 1092)

Changing a password (Page 1091)

9.1.7.4 Opening know-how protected blocks

You can only open multiple know-how protected blocks at once if they are protected with the same password.

Procedure

To open a know-how protected block, follow these steps:

1. Double-click on the block you wish to open.

   The "Access protection" dialog will open.
2. Enter the password for the know-how protected block.
3. Confirm your entry with "OK".

Result

The know-how protected block will open provided you have entered the correct password. However, the block will remain know-how protected. If you copy the block or add it to a library, for example, the copies will also be know-how protected.

Once you have opened the block, you can edit the program code and the block interface of the block for as long as the block or TIA portal is open. The password must be entered again the next time the block is opened. If you close the "Access protection" dialog with "Cancel", the block will open but the block code will not be displayed and you will not be able to edit the block.

See also

Protecting blocks (Page 1085)

Setting up and removing block copy protection (Page 1087)

Setting up block know-how protection (Page 1088)

Printing know-how protected blocks (Page 1090)

Removing block know-how protection (Page 1092)

Changing a password (Page 1091)

9.1.7.5 Printing know-how protected blocks

You can only print complete know-how protected blocks if they have been opened with the correct password. If you print a closed block or if the block was not opened with the correct password, only the non-protected block data will be printed.

Procedure

To print a know-how protected block in full, follow these steps:

1. Open the know-how protected block you wish to print.

   See also: Opening know-how protected blocks (Page 1089)
2. Select the "Print" command in the "Project" menu.

   The "Print" dialog will open.
3. Select the printer in the "Name" field.
4. Click "Advanced" to modify the Windows printer settings.
5. Select the documentation information set in the "Document information" drop-down list that you want to use for the frame layout.
6. Under "Print objects/area" select whether you want to print all objects or the complete area, or only a selection.
7. Under "Properties" select the print scope.
   - Select "All" to print the complete block.
   - Choose "Visible" to print all the information within the block that is visible on the screen.
   - Select "Compact" to print a shortened form of the block.
8. Click "Preview" to generate a print preview in advance.

   A print preview is created in the work area.
9. Click "Print" to start the printout.

See also

Protecting blocks (Page 1085)

Setting up and removing block copy protection (Page 1087)

Setting up block know-how protection (Page 1088)

Removing block know-how protection (Page 1092)

Changing a password (Page 1091)

9.1.7.6 Changing a password

Procedure

To change the password, follow these steps:

1. Select the know-how protected blocks for which you want to change the password.

   Note

   You can only change the password for several blocks at once if all blocks selected have the same password.

2. Select the command "Know-how protection" in the "Edit" menu.

   The "Know-how protection" dialog will open.

3. Click the "Change" button.
4. Enter the old password in the "Old" field.
5. Enter the new password in the "New" field.
6. Enter the new password again in the "Confirm" field.
7. Confirm your entries with "OK".
8. Close the "Know-how protection" dialog by clicking on "OK".

See also

Protecting blocks (Page 1085)

Setting up and removing block copy protection (Page 1087)

Setting up block know-how protection (Page 1088)

Opening know-how protected blocks (Page 1089)

Printing know-how protected blocks (Page 1090)

Removing block know-how protection (Page 1092)

9.1.7.7 Removing block know-how protection

Procedure

To remove block know-how protection, follow these steps:

1. Select the blocks for which you want to remove know-how protection.

   Note

   You can only remove know-how protection for several blocks at once if all blocks selected have the same password.

2. Select the command "Know-how protection" in the "Edit" menu.

   The "Know-how protection" dialog will open.
3. Deactivate the check box "Hide code (know-how protection)".
4. Enter the password.
5. Confirm your entries with "OK".

Result

Know-how protection will be disabled for the blocks selected.

See also

Protecting blocks (Page 1085)

Setting up and removing block copy protection (Page 1087)

Setting up block know-how protection (Page 1088)

Opening know-how protected blocks (Page 1089)

Printing know-how protected blocks (Page 1090)

Changing a password (Page 1091)

9.2 Displaying program information

9.2.1 Overview of available program information

Program information

The program information of a user program contains the view specified in the following table.

View	Application
Assignment list (Page 1094)	Provides an overview of the address bits for the I, Q, and M memory areas already allocated within the user program. Also indicates if an address has been allocated by access from an S7 program or if the address has been assigned to a SIMATIC S7 module.
Call structure (Page 1103)	Shows the call structure of the blocks within the user program and provides an overview of the blocks used and their relationships.
Dependency structure (Page 1111)	Shows the list of blocks used in the user program. A block is shown at the first level and blocks that call or use this block are indented below it. In contrast to the call structure, instance blocks are listed separately.
Resources (Page 1117)	Shows the hardware resources of the CPU for objects (OB, FC, FB, DB, PLC tags and user-defined data types), for CPU memory areas and for the existing I/O modules.

Displaying several views simultaneously

You can generate and display several views for one or more user programs to facilitate testing and changing your user program.

Displaying multiple views, for example, enables you to:

- Display all program information for a user program next to one another
- Compare different user programs

9.2.2 Displaying an assignment list

9.2.2.1 Introduction to the assignment list

Program information in the assignment list

The assignment list shows if an address has been allocated by access from an S7 program or if the address has been assigned to a SIMATIC S7 module. It is therefore an important basis for locating errors or changes in the user program.

In the assignment list, you have a CPU-specific overview of which bit is used in which byte of the memory areas listed below:

- Input (I)
- Output (O)
- Bit memory (M)
- Timer (T)
- Counter (C)
- I/O (P)

Display of the assignment list

The assignment list of inputs, outputs, and bit memory is displayed in several separate work windows.

Filters

You can filter the display within the assignment list. You can use predefined filters or create your own.

Displaying cross-reference information

You have the option of displaying cross-reference information for selected addresses in the assignment list.

You can display the cross-references for a selected address in the Inspector window using the "Cross-reference information" shortcut menu command. The command "Tools > Cross-references" allows you to also open the cross-reference list for the selected object.

Displaying the PLC tag table

You can open the PLC tag table from the assignment list and edit the properties of the tags used.

To do this select an address of the assignment list and select the "Open editor" command in the shortcut menu.

Enabling the display of retentivity

You can enable and disable the display of the retentive state of bit memory by selecting the "Hide/show retain area" toolbar button.

See also

Symbols in the assignment list (Page 1096)

Layout of the assignment list (Page 1095)

9.2.2.2 Layout of the assignment list

Layout of the assignment list

Depending on the CPU, the assignment list is displayed in several work windows with the following operands.

For S7-300/400 CPUs:

- Inputs
- Outputs
- Bit memory
- Timers
- Counters

For S7-1200 CPUs:

- Inputs
- Outputs
- Bit memory

Displaying inputs, outputs, bit memory, timers and counters

It shows all operands used and their assignment in the S7 program.

For all displayed operands, each line in the assignment list is dedicated to a byte of the memory area, in which the corresponding eight bits from 7 to 0 are labeled according to their access. In conclusion, a "bar" indicates if access is made by a byte (B), word (W) or double word (D).

You can find an explanation of the symbols in the assignment list here. (Page 1096)

See also

Introduction to the assignment list (Page 1094)

9.2.2.3 Symbols in the assignment list

Meaning of the symbols in the assignment list

The following table shows the meaning of the symbols in the assignment list:

Symbol	Meaning
◆	Indicates the address assignment in the selected state.
◆	Indicates the address assignment in the non-selected state.
◇	Indicates that a pointer start address and a tag address access the same address range and that they are selected.
◇	Indicates that a pointer start address and a tag address access the same address range and that they are not selected.
•	Indicates the pointer assignment in the selected state.
•	Indicates the pointer assignment in the non-selected state.
▮	Indicates that the byte is in use with byte access and the corresponding tag is selected. The shortcut menu allows you to display cross-reference information for the selected variables as well as the PLC tag table.
▮	Indicates that the byte is in use with byte access and the corresponding tag is not selected.
▮ ▮	Indicates that the byte is in use with word access and the corresponding tag is selected. The shortcut menu allows you to display cross-reference information for the selected variables as well as the PLC tag table.
▮ ▮	Indicates that the byte is in use with word access and the corresponding tag is not selected.
▮ ▮ ▮ ▮	Indicates that the byte is in use with double word access and the corresponding tag is selected. The shortcut menu allows you to display cross-reference information for the selected variables as well as the PLC tag table.
▮ ▮ ▮ ▮	Indicates that the byte is in use with double word access and the corresponding tag is not selected.
Background color: gray	Indicates that a byte is in use with byte, word or double word access and that the address is also in use by the hardware. The gray background color indicates overlapping memory access.
Background color: yellow	Indicates that the address is not in use by the hardware.
▦	Indicates that the memory area has been defined as system memory.
⏲	Indicates that the memory area has been defined as clock memory.

See also

Layout of the assignment list (Page 1095)

Introduction to the assignment list (Page 1094)

9.2.2.4 Displaying an assignment list

Requirement

A project has been created with programmed blocks.

Procedure

Proceed as follows to display the assignment list:

1. Select the "Program blocks" folder or one or more of the blocks it contains.
2. Select the "Assignment list" command in the "Tools" menu.

Result

The assignment list for the selected program is displayed.

View options in the assignment list

Refer to view respective view options that are set to display the desired information in the assignment list.

See also

Setting the view options for the assignment list (Page 1098)

Layout of the assignment list (Page 1095)

9.2.2.5 Setting the view options for the assignment list

Introduction

The following view options are available for the assignment list:

- Used addresses:

  When this check box is activated, the addresses, I/Os and pointers used in the program are displayed.

- Free hardware addresses:

  When this check box is activated, only the free hardware addresses are displayed.

Requirement

- A project has been created with programmed blocks.
- The assignment list is open.

Procedure

Proceed as follows to set the view options for the assignment list:

1. Click on the arrow of the symbol ("View options") in the task bar.

   The view options for the assignment list are opened. Check marks are set in front of the activated view options.

2. If you want to activate or deactivate a view option, click on the respective check box and set or remove the check mark.

Result

The view options are set and the desired information is displayed in the assignment list.

9.2.2.6 Filter options in the assignment list

Filter settings

You can define your own filter settings for the assignment list. The following options are available for defining filters:

- Display all addresses of the address areas specified.
- Display of single, defined addresses from the selected address area, for example, "0" and "200".
- Display of complete areas from the selected address area, for example, "0 - 256".

The following table provides an overview of all available options:

Selection in the	Selection	Symbol	Meaning
Address area	All CPU-dependent displayed addresses (I, O, M, T, C) can be activated as they are by default, or individual address areas can be activated.	Check box is activated	Only the activated address areas (I, O, M, T, C) are shown in the assignment list.
Filter area	Show assignment for all addresses	*	Displays the assignment of all addresses of the enabled address areas (I, Q, M).
	Show assignment for selected addresses, for example, for the inputs "IB 0" and "IB 256"	0;256 Separate individual addresses and areas by a semicolon.	Assignments of selected addresses for the activated address areas (I) are shown.
	Show assignment for selected areas, for example, for the inputs "IB 0 to IB 100" and "IB 200 to IB 256".	0-100;200-256 Contiguous areas should be connected by a hyphen.	Assignments of selected areas for the activated address areas (I) are shown.

9.2.2.7 Defining filters for assignment list

Requirement

- A project has been created with programmed blocks.
- The assignment list is open.

Defining filter

Proceed as follows to define a filter for the assignment list:

1. Click on the symbol ("Filter") in the task bar.

   The "Assignment List Filter" dialog opens.
2. Click on the symbol ("Create new filter") in the task bar.

   A new filter is created with the name "Filter_1". The check boxes for all addresses (inputs, outputs, memory bits, timers and counters) are activated by default for the filter.
3. If you want to change the name of the filter, click on the drop-down list in the task bar and enter a new filter name.
4. Deactivate the check boxes of addresses that are not to be affected by the filter.
5. Enter one of the following options in the filter area of the activated address:
   - Show all addresses used = "*"
   - Show single, defined addresses, for example, IB 0" and IB 25 = "0.25". Individual addresses and address areas are separated by commas or semicolons.
   - Show complete address areas, for example, IB 0 to IB 256 = "0-256". Complete address areas should be connected by a hyphen.
6. Confirm your entries with "OK".

   The newly defined filter is shown in the task bar of the assignment list under the specified name.

Delete filter

Proceed as follows to delete a filter:

1. Click on the symbol ("Filter") in the task bar.

   The filter dialog for the assignment list opens.
2. In the drop-down list of the task bar, select the filter you want to delete.
3. Click on the symbol ("Delete selected filter") in the task bar.

   The selected filter is deleted.

See also

Filter options in the assignment list (Page 1099)

Displaying an assignment list (Page 1097)

Introduction to the assignment list (Page 1094)

9.2.2.8 Filtering an assignment list

Requirement

- A project has been created with programmed blocks.
- The assignment list is open.

Procedure

1. Click on the arrow on the drop-down list.

   The available filter are displayed.
2. Select the desired filter.

Result

The assignment list is filtered according to the settings of the selected filter.

Note

The filter settings are saved when the project is closed.

9.2.2.9 Defining retentive memory areas for bit memories

Introduction

In the assignment list you can define the width of the retentive memory area for bit memories. The content of tags which are addressed in retentive memory is retained after power off and at the STOP to RUN transition after power on.

The display of retentive bit memories can be enabled and disabled in the assignment list. If their display is enabled, retentive bit memories are identified by an icon in the "Address" column.

Requirement

The assignment list is open.

Procedure

Proceed as follows to define the width of the retentive memory area for bit memories:

1. Click "Retain" in the toolbar.

   The "Retain memory" dialog will open.
2. Starting at the count of 0, define the width of the retentive memory area by entering the last byte of this area in the input field. Watch out for any addresses of tags already assigned to the retentive area.
3. Load the block to the target system. Select the "Program blocks" folder in the Project tree and select the "Download to device" submenu in the shortcut menu.

Result

The width of the retentive memory area is defined. If enabled in the assignment list, an icon will indicate the retentive state of all tags in the "Address" column.

9.2.2.10 Enabling the display of retentive bit memories

Introduction

In the assignment list you can enable and disable the display of retentive bit memories. The retentive bit memories are identified by means of an icon in the "Address" column if the display of retentivity is enabled.

Requirement

The assignment list is open.

Procedure

Proceed as follows to enable and disable the display of retentive bit memories:

1. Click "Display/hide retentivity" in the toolbar.

Result

The retentive tags are identified by means of an icon in the "Address" column of the bit memory area if the display of retentivity is enabled. The icons in the "Address" column are hidden if the display of retentivity is disabled.

9.2.3 Displaying the call structure

9.2.3.1 Introduction to the call structure

Call structure

The call structure describes the call hierarchy of the block within an S7 program.

It provides an overview of:

- The blocks used
- Jumps to the places of use of the blocks
- Relationships between blocks
- Local data requirements of the blocks
- Status of the blocks

Information in the call structure

Displaying the call stucture provides you with a list of the blocks used in the user program. The first level of the call structure is highlighted in color and shows the blocks that are not called by any other block in the program. Organization blocks are always shown on the first level of the call structure. Functions, function blocks and data blocks are only shown on the first level if they are not called by an organization block. When a block calls other blocks or functions, they are listed indented under the calling block. Instructions and blocks are shown in the call structure only if they are called by a block.

View options

The following view options are available for the call structure:

- Show conflicts only:

  When this check box is activated, only the conflicts within the call structure are displayed.
- Group multiple calls together:

  When this check box is activated, several block calls are grouped together. The number of block calls is displayed in the "Call frequency" column. The links to the various call locations are offered in a drop-down list in the "Details" column.

Displaying the block calls

You can display the block calls in a block by clicking on the arrow in front of the block title. To display the call information of all blocks, click on the "Expand list" icon in the toolbar.

You can hide the total overview by clicking the "Collapse list" icon.

Displaying cross-reference information

You can display the cross-reference information for a block in the Inspector window by right-clicking on the relevant block and selecting the "Cross-reference information" command from the shortcut menu.

To open the "Cross-references" view, click the "Cross-references" command in the shortcut menu.

Displaying blocks in the program editor

You can open the program editor and edit blocks there from the call structure.

To do this select the required block in the call structure and select the "Open editor" command in the shortcut menu.

Displaying deleted blocks

The rows belonging to deleted blocks are identified by an icon.

Note

Please note that any existing local data can only be displayed or updated after compiling a block.

See also

Symbols in the call structure (Page 1105)

9.2.3.2 Symbols in the call structure

Meaning of the symbols in the call structure

The following table shows the meaning of the symbols in the call structure:

Symbol	Meaning
🟪	Indicates an organization block (OB).
🟦	Indicates a function block (FB).
🟩	Indicates a function (FC).
⛁	Indicates a data block (DB).
⛁	Indicates that the block is declared as a multiinstance.
⊡	The object has an interface dependency to an object connected to the left.
□	Indicates that the block needs to be compiled again.
▯	Indicates that the data block needs to be compiled again.
╲	Indicates that the object is not available.
⊐	Indicates that the interface causes a time stamp conflict.
🕓	Indicates that the variable causes a time stamp conflict.
▪	Indicates that the block is not called directly or indirectly from an OB.
🔒	Indicates that an object has know-how protection.
▶	Indicates that the block is normally called recursively.
⊡	Indicates that a tag declaration in the interface has a recursive dependency: • Scenario 1: FB1 calls FB2 and this then calls FB1. The instance data blocks of these FBs have a recursion in the interface. • Scenario 2: A multiple instance FB uses the instance DB of its parent FB as a global DB.

9.2.3.3 Layout of the call structure

Layout of the call structure

The view of the call structure consists of the following columns:

Column	Content/meaning
Call structure	Shows an overview of the blocks called If the viewing option "Group multiple calls together" is enabled, several block calls are grouped together and the "Number of calls" column is displayed.
Call type (!)	Shows the type of call, for example recursive block call.
Address	Shows the absolute address of the block. With a function block, the absolute address of the corresponding instance data block is also shown.
Details	Shows the network or interface of the calling block. All information are offered as a link in this column. With this link, you can jump to the location of the block call in the program editor. If the viewing option "Group multiple calls together" option is enabled, the calls are grouped together and are available as links in a drop-down list.
Local data (in path)	Indicates the local data requirement of the full path. Blocks with optimized access have higher local data requirements because the information for the symbolic addressing is stored with them. Please note that any existing local data can only be displayed or updated after compiling a block.
Local data (for blocks)	Show the local data requirements of the block. Blocks with optimized access have higher local data requirements because the information for the symbolic addressing is stored with them. Please note that any existing local data can only be displayed or updated after compiling a block.

See also

Symbols in the call structure (Page 1105)

Introducing the consistency check in the call structure (Page 1109)

9.2.3.4 Displaying the call structure

Requirement

A project has been created with blocks.

Procedure

Proceed as follows to display the call structure:

1. Select the "Program blocks" folder or one or more of the blocks it contains.
2. Select the "Call structure" command in the "Tools" menu.

Result

The call structure for the selected program is displayed.

Note

Please note that any existing local data can only be displayed or updated after compiling a block.

See also

Setting the view options for the call structure (Page 1108)

9.2.3.5 Setting the view options for the call structure

Introduction

The following view options are available for the call structure:

- Show conflicts only:

  Only the blocks causing conflicts within the call structure are displayed if this check box is activated.

  The following blocks cause conflicts:

  – Blocks executing any calls with older or newer code time stamps.

  – Blocks calling a block with modified interface.

  – Blocks using a tag with modified address and/or data type.

  – Block called neither directly, nor indirectly by an OB.

  – Blocks calling a block which no longer exists.

- Group multiple calls together:

  When this viewing option is enabled, several block calls and data block accesses are grouped together. The number of block calls is displayed in the "Call frequency" column. The links to the various call locations are offered in a drop-down list in the "Details" column.

Requirement

- A project has been created with programmed blocks.
- The call structure is open.

Procedure

Proceed as follows to set the view options for the call structure:

1. Click on the arrow of the symbol ("View options") in the task bar.

   The view options for the call structure opens. Check marks are set in front of the activated view options.

2. If you want to activate or deactivate a view option, click on the respective check box and set or remove the check mark.

Result

The view options are set and the required information is displayed in the call structure.

9.2.3.6 Introducing the consistency check in the call structure

Consistency check

Changing the time stamp of a block during or after the program is generated can lead to time stamp conflicts, which in turn cause inconsistencies among the blocks that are calling and being called.

Using the consistency check

The "Consistency check" function is used to visualize inconsistencies when time stamp conflicts occur. Whe the consistency check is performed, the inconsistent blocks are shown in the call structure and marked with the correspoinding symbols.

- Most time stamp and interface conflicts can be rectified by recompiling the blocks.
- If compilation fails to clear up inconsistencies you can use the link in the "Details" column to go to the source of the problem in the program editor and manually eliminate any inconsistencies.
- The blocks marked in red must be recompiled.

See also

Symbols in the call structure (Page 1105)

9.2.3.7 Checking block consistency in the call structure

Requirement

- A project has been created with programmed blocks.
- The call structure is open.

Procedure

Proceed as follows to check the block consistency:

1. Click on the symbol ("Consistency check") in the task bar.

   The block consistency is checked. Blocks found to be inconsistent are marked accordingly by a symbol.
2. If a block is inconsistent, click on the arrow in front of the block title in the call structure.

   The inconsistent blocks are displayed. The exact problem locations are listed as links in the "Details" column.
3. Click on the respective link in the "Details" column to jump to the location in the block requiring correction.
4. Check and correct the inconsistencies in the blocks.
5. Recompile the blocks by selecting the required blocks and clicking on the command "Compile" in the shortcut menu.
6. Download the corrected blocks to the target system by clicking the command "Download to device" in the shortcut menu.

Result

The block consistency is checked. The inconsistencies in the blocks are corrected. The corrected blocks are loaded to the target system.

See also

Symbols in the call structure (Page 1105)

9.2.4 Displaying the dependency structure

9.2.4.1 Introduction to the dependency structure

Introduction

The dependency structure shows the dependencies each block has to other blocks in the program.

Information in the dependency structure

Displaying the dependency structure provides you with a list of the blocks used in the user program. A block is shown at the far left and blocks that call or use this block are indented below it.

The dependency structure also shows the status of the individual blocks using symbols.

Objects causing a time stamp conflict and perhaps leading to an inconsistency in the program are marked with various symbols.

The dependency structure is an extension of the cross-reference list for objects.

View options

The following view options are available for the dependency structure:

- Show conflicts only:

  When this check box is activated, only the conflicts within the dependency structure are displayed.

- Group multiple calls together:

  When this check box is activated, several block calls are grouped together. The number of block calls is shown numerically in the "Dependency structure" column. The links to the various call locations are offered in a drop-down list in the "Details" column.

Displaying the dependency structure

Clicking on the arrow in front of the block title displays the blocks that call or use this block. To display the dependencies of all blocks,

click the "Expand list" icon in the toolbar.

You can hide the total overview by clicking the "Collapse list" icon.

Displaying cross-reference information

You can display the cross-reference information for a block in the Inspector window by right-clicking on the respective block and selecting the "Display Usage" command from the shortcut menu.

Displaying blocks in the program editor

You can open the program editor and edit blocks there from the dependency structure. To do this select the required block in the dependency structure and select the "Open editor" command in the shortcut menu.

9.2.4.2 Layout of the dependency structure

Layout of the dependency structure

The view of the dependency structure consists of the following columns:

Column	Content/meaning
Dependency	It indicates the dependencies between each block and the other blocks in the program.
Call type (!)	Shows the type of call, for example recursive block call.
Address	Shows the absolute address of the block.
Call frequency	Indicates the number of multiple calls of blocks.
Details	Shows the network or interface of the called block. All information are offered as a link in this column. With this link, you can jump to the location of the block call in the program editor. If the viewing option "Group multiple calls together" option is enabled, the calls are grouped together and are available as links in a drop-down list.

See also

Symbols in the dependency structure (Page 1113)

9.2.4.3 Symbols in the dependency structure

Meaning of the symbols in the dependency structure

The following table shows the meaning of the symbols in the dependency structure:

Symbol	Meaning
🟪	Indicates an organization block (OB).
🟦	Indicates a function block (FB).
🟩	Indicates a function (FC).
⛁	Indicates a data block (DB).
⊡	The object has an interface dependency to an object connected to the left.
□	Indicates that the block needs to be compiled again.
⬯	Indicates that the data block needs to be compiled again.
🕒	Indicates that there is an inconsistency with this object.
🔒	Indicates that an object has know-how protection.
⊡	Indicates that a tag declaration in the interface has a recursive dependency: • Scenario 1: FB1 calls FB2 and this then calls FB1. The instance data blocks of these FBs have a recursion in the interface. • Scenario 2: A multiple instance FB uses the instance DB of its parent FB as a global DB.

9.2.4.4 Displaying the dependency structure

Requirement

A project has been created with programmed blocks.

Procedure

Proceed as follows to display the dependency structure:

1. Select the block folder or one or more of the blocks contained therein.
2. Select the "Dependency structure" command in the "Tools" menu.

Result

The dependency structure for the selected program is displayed.

See also

Setting the view options for the dependency structure (Page 1114)

9.2.4.5 Setting the view options for the dependency structure

Introduction

The following view options are available for the dependency structure:

- Show conflicts only:

  When this check box is activated, only the conflicts within the dependency structure are displayed.

  The following blocks cause conflicts:

  – Blocks executing any calls with older or newer code time stamps.

  – Blocks called by a block with modified interface.

  – Blocks using a tag with modified address and/or data type.

  – Block called neither directly, nor indirectly by an OB.

- Group multiple calls together:

  When this check box is activated, several block calls are grouped together. The number of block calls is shown in the relevant column. The links to the various call locations are offered in a drop-down list in the "Details" column.

Requirement

- A project has been created with programmed blocks.
- The dependency structure is open.

Procedure

Proceed as follows to set the view options for the dependency structure:

1. Click on the arrow of the symbol ("View options") in the task bar.

   The view options for the dependency structure are opened. Check marks are set in front of the activated view options.

2. If you want to activate or deactivate a view option, click on the respective check box and set or remove the check mark.

Result

The view options are set and the required information is displayed in the dependency structure.

9.2.4.6 Introducing the consistency check in the dependency structure

Consistency check

Changing the time stamp of a block during or after the program is generated can lead to time stamp conflicts, which in turn cause inconsistencies among the blocks that are calling and being called.

Using the consistency check

The "Consistency check" function is used to visualize inconsistencies. Whe the consistency check is performed, the inconsistent blocks are shown in the dependency structure and marked with the correspoinding symbols.

- Most time stamp and interface conflicts can be rectified by recompiling the blocks.
- If compilation fails to clear up inconsistencies you can use the link in the "Details" column to go to the source of the problem in the program editor and manually eliminate any inconsistencies.
- The blocks marked in red must be recompiled.

See also

Layout of the dependency structure (Page 1112)

Symbols in the dependency structure (Page 1113)

9.2.4.7 Checking block consistency in the dependency structure

Requirement

- A project has been created with programmed blocks.
- The dependency structure is open.

Procedure

Proceed as follows to check the block consistency:

1. Click on the symbol ("Consistency check") in the task bar.

   The block consistency is checked. Blocks found to be inconsistent are marked accordingly by a symbol.
2. If a block is inconsistent, click on the arrow in front of the block title in the dependency structure.

   The inconsistent blocks are displayed. The exact problem locations are listed as links in the "Details" column.
3. Check and correct the inconsistencies in the blocks.
4. Recompile the blocks by selecting the required blocks and clicking on the command "Compile" in the shortcut menu.
5. Download the corrected blocks to the target system by clicking the command "Download to device" in the shortcut menu.

Result

The block consistency is checked. The inconsistencies in the blocks are corrected. The corrected blocks are loaded to the target system.

See also

Symbols in the dependency structure (Page 1113)

9.2.5 Displaying CPU resources

9.2.5.1 Introducing resources

Introduction

The "Resources" tab indicates the hardware resources of the configured CPU for:

- the used programming objects,
- the assignment of memory areas within the CPU and
- the assigned inputs and outputs of the existing input and output modules.

Information provided in the "Resources" tab

The resources tab provides an overview of the hardware resources used on your CPU for:

- the programming objects used in the CPU (e.g. OB, FC, FB, DB, PLC tags, and user-defined data types),
- the memory areas available on the CPU (work memory, load memory, retentive memory), their maximum size and utilization by the programming objects stated above,
- the I/O of modules which can be configured for the CPU (I/O modules, digital input modules, digital output modules, analog input modules, and analog output modules), including the I/O already in use.

Display of the maximum available load memory

The maximum size of available load memory can be selected from a drop-down list box in the "Total" row of the "Load memory" column.

Display of the maximum available work memory

The maximum size of available work memory can be selected from a drop-down list box in the "Total" row of the "Work memory" column.

Display of the maximum available retentive memory

The maximum size of available retentive memory can be selected from a drop-down list box in the "Total" row of the "Retentive memory" column.

Note

Retentive memory data

All bit memories and data blocks specified as retentive will be integrated in the calculation of the retentive data.

Updating the display in the "Resources" tab

Click the "Update view" toolbar button to update the display of objects.

Benefits of the display in the "Resources" tab

The "Resources" tab of the program information dialog provides a detailed list of all objects and of the corresponding memory area used.

The tab also indicates shortage of resources and helps to avoid such states.

Blocks which are not compiled can be identified as their size is indicated by a question mark.

See also

Layout of the "Resources" tab (Page 1119)

Displaying resources (Page 1120)

Selecting the maximum load memory available (Page 1120)

9.2.5.2 Layout of the "Resources" tab

Layout of the "Resources" tab in the program information

The view of the "Resources" tab consists of the following columns:

Column	Content/meaning
Objects	The "Details" area provides an overview of the programming objects available in the CPU, including their memory assignments.
Load memory	Displays the maximum load memory resources of the CPU as a percentage and as absolute value. The values displayed under "Total" provide information on the maximum memory available in the load memory. The values displayed under "Used" provide information on the memory actually used in the load memory.
Work memory	Displays the maximum work memory resources of the CPU as a percentage and as absolute value. The values displayed under "Total" provide information on the maximum memory available in the work memory. The values displayed under "Used" provide information on the memory actually used in the load memory.
Retentive memory	Displays the maximum resources for retentive memory in the CPU as a percentage and as absolute value. The values displayed under "Total" provide information on the maximum memory available in the retentive memory. The values displayed under "Used" provide information on the memory actually used in the load memory.
I/O	Displays the I/Os which are available on the CPU, including their module-specific availability in the next columns. The values displayed at "Configured" provide information about the maximum number of I/O available. The values displayed under "Used" provide information on the memory actually used in the load memory.

DI / DQ / AI / AQ	Displays the number of configured and used inputs/outputs:
	DI = Digital inputs
	DQ = Digital outputs
	AI = Analog inputs
	AQ = Analog outputs
	The values displayed at "Configured" provide information about the maximum number of I/O available.
	The values displayed under "Used" provide information on the actually used inputs and outputs.

See also

Displaying resources (Page 1120)

Selecting the maximum load memory available (Page 1120)

Introducing resources (Page 1117)

9.2.5.3 Displaying resources

Requirement

A project with programmed blocks has been created.

Procedure

Proceed as follows to display the resources of the respective CPU memory areas:

1. Select the block folder or one or several of the blocks contained therein.
2. Select the "Resources" command in the "Tools" menu.

Result

The memory resources of the CPU are displayed.

9.2.5.4 Selecting the maximum load memory available

Requirement

A project with programmed blocks has been created.

Procedure

Proceed as follows to display the available maximum of load memory resources:

1. Open the drop-down list in the "Total" field of the "Load memory" column by clicking the icon.
2. Select a corresponding value for the CPU used by clicking it in the drop-down list box.

Result

The "Total" field displays the selected maximum memory resources.

9.3 Displaying cross-references

9.3.1 General information about cross references

Introduction

The cross-reference list provides an overview of the use of operands and tags within the user program.

Uses of cross-references

The cross-reference list offers you the following advantages:

- When creating and changing a program, you retain an overview of the operands, tags and block calls you have used.
- From the cross-references, you can jump directly to the point of use of operands and tags.
- During a program test or when troubleshooting, you are informed of the following:
  - which operand is processed by which command in which block,
  - which tag is used in which picture,
  - which block is called by which other block.
- As part of the project documentation, the cross-references provide a comprehensive overview of all operands, memory areas, blocks, tags and pictures used.

See also

Structure of the cross-reference list (Page 1122)

Displaying the cross-reference list (Page 1124)

Displaying cross-references in the Inspector window (Page 1125)

9.3.2 Structure of the cross-reference list

Views of the cross-reference list

There are two views of the cross-reference list. The difference between the two views is in the objects displayed in the first column:

- Used by:

  Display of the referenced objects. Here, you can see where the object is used.
- Used:

  Display of the referencing objects. Here, you can see the users of the object.

The assigned tool tips provide additional information about each object.

Structure of the cross-reference list

The cross-reference list has the following structure:

Column	Content/meaning
Object	Name of the object that uses the lower-level objects or that is being used by the lower-level objects.
Number	Number of uses
Point of use	Each point of use, for example, network
Property	Special properties of referenced objects, for example, the tag names in multi-instance declarations.
as	Shows additional information about the object, e.g., that an instance DB is used as template or as multiple instance.
Access	Type of access, whether access to the operand is read access (R) and/or write access (W).
Address	Address of the operand
Type	Information on the type and language used to create the object
Path	Path of object in project tree

Depending on the installed products, additional columns or different columns are displayed for the cross-references.

Settings in the cross-reference list

You can make the following settings using the buttons in the toolbar of the cross-reference list:

- Update cross-reference list

  Updates the current cross-reference list.

- Making settings for the cross-reference list

  Here, you select check boxes to specify whether all used, all unused, all defined or all undefined objects will be displayed. If the "Undefined objects" option is enabled, references to previously deleted objects are also displayed.

- Collapse entries

  Reduces the entries in the current cross-reference list by closing the lower-level objects.

- Expand entries

  Expands the entries in the current cross-reference list by opening the low-level objects.

Sorting in the cross-reference list

You can sort the entries in the "Object" column, including other product-specific columns, in ascending or descending order. To do this, click on the relevant column title.

See also

General information about cross references (Page 1121)

Displaying the cross-reference list (Page 1124)

9.3.3 Displaying the cross-reference list

Prerequisites

You have created a project.

Introduction

There are several ways of displaying cross-references depending on whether you are in the Portal view or in the Project view and which object you have selected in the project tree.

In the Portal view, you can only display cross-references for the entire CPU; in the Project view, you can, for example, display cross-references for the following objects:

- "PLC tags" folder
- "PLC data types" folder
- "Program blocks" folder
- "Tags and connections" folder
- Individual tags
- Individual PLC data types
- Individual blocks
- Technological objects

Displaying cross-references

Proceed as follows to display cross-references:

1. Select the required action in the Portal view, for example "PLC programming" and the "Show cross-references" command or select one of the objects listed above in the Project view and select the "Cross-references" command in the "Tools" menu.
2. Click the "Used by" button to display where the objects shown in the cross-reference list are used.
3. Click the "Uses" button to view the users of the objects displayed in the cross-reference list.
4. You can perform the following actions using the buttons in the toolbar:
   - Update cross-reference list
   - Making settings for the cross-reference list
   - Collapse entries
   - Expand entries
5. You can sort the entries in the "Object" and "Address" columns in ascending or descending order by clicking on the relevant column title.
6. To go to the point of use of the object, click on the displayed link.

See also

General information about cross references (Page 1121)

Structure of the cross-reference list (Page 1122)

9.3.4 Displaying cross-references in the Inspector window

Introduction

The Inspector window displays cross-reference information about an object you have selected in the "Info > Cross-references" tab. This tab displays the instances where a selected object is being used and the other objects using it.

The Inspector window also includes blocks which are only available online in the cross-references.

Structure

The Inspector window displays the cross-reference information in tabular format. Each column contains specific and detailed information on the selected object and its application. The table below shows the additional information listed in the "Info > Cross-reference" tab:

Column	Meaning
Object	Name of the object that uses the lower-level objects or that is being used by the lower-level objects.
Number	Number of uses
Point of use	Each location of use, for example, network
Property	Special properties of referenced objects, for example, the tag name in multi-instance declarations
as	Shows additional information about the object, e.g., that an instance DB is used as template or as multiple instance.
Access	Access mode Shows whether the operand is accessed by a read (R) and/or write (W) operation.
Address	Address of the operand
Monitor value	This column will only be displayed when the program editor is open. Here you can monitor global tabs using the shortcut menu.
Type	Information about the type and language used to create the object
Path	Path of object in project tree

Depending on the installed products, additional columns or different columns are displayed for the cross-references.

9.4 Testing the user program

9.4.1 Basics of testing the user program

Functions

You have the option of testing the running of your user program on the device. You can then monitor signal states and values of tags and can assign values to tags to simulate certain situations in the running of the program.

Requirement

There must be an executable program loaded on the device.

Test options

The following test options are available:

- Testing with program status

  The program status allows you to monitor the running of the program. You can display the values of operands and the results of logic operations (RLO) allowing you to recognize and fix logical errors in your program.
- Testing in single step mode (S7-300/400 only)

  You can test blocks you created in STL or SCL in the single step mode. You do this by setting breakpoints in the program code at which program execution stops. You can then continue to run the program one step at a time. Within a CPU, you can test either with program status or in single step mode. You cannot, however, use both test options at the same time within a CPU.
- Testing with the watch table

  With the watch table, you can monitor and modify the current values of individual tags in the user program or on a CPU. You can assign values to individual tags for testing and run the program in a variety of different situations. You can also assign fixed values to the I/O outputs of a CPU in STOP mode, for example to check the wiring.
- Testing with the force table

  With the force table, you can monitor and force the current values of individual tags in the user program or on a CPU. When you force, you overwrite individual tags with specified values. This allows you to test your user program and run through various situations.

  When forcing, make sure that you keep to the necessary safety measures for forcing (Page 1173)!

See also

Introduction to testing with program status (Page 1128)

Introduction to testing with the watch table (Page 1133)

Introduction for testing with the force table (Page 1156)

9.4.2 Testing with program status

9.4.2.1 Testing the program

Introduction to testing with program status

Function

If you display the program status, you can monitor the execution of the program. This provides you with an overview of the values of the individual operands and the results of the logic operations and you can check whether the components of the automation system are correctly controlled.

⚠ WARNING

Testing while the plant is operating can cause serious damage to property or injury to persons if there are functional disturbances or program errors.

Make sure that no dangerous situations can arise before you conduct a test.

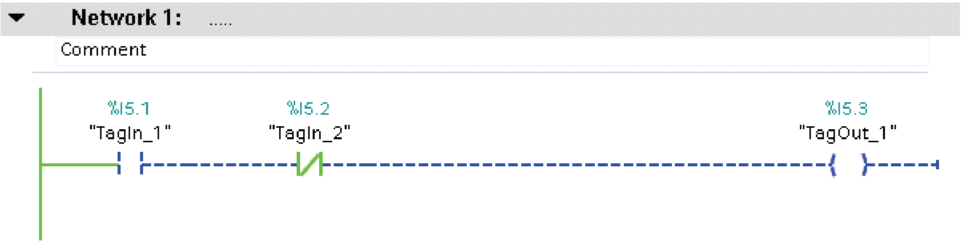
Displays in program status

Program status display for LAD programs

Displays in program status

The display of the program status is updated cyclically.

The following figure shows an example of the program status display for LAD:



Representation of the program status

You can recognize the status of individual instructions and lines of a network quickly based on the color and type of lines and symbols. The following table shows the relationship between representation and status:

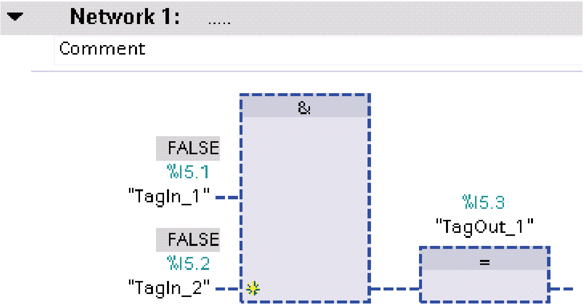
Representation	Status
Green solid	Satisfied
Blue dashed	Not satisfied
Gray solid	Unknown or not executed
Black	Not interconnected
Parameter in a frame with a saturation of 100 %	Value is current
Parameter in a frame with a saturation of 50 %	Value originates from an earlier cycle. The point in the program was not executed in the current cycle.

Program status display for FBD programs

Displays in program status

The display of the program status is updated cyclically.

The following figure shows an example of the program status display for FBD:



Representation of the program status

You can recognize the status of individual instructions and lines of a network quickly based on the color and type of lines and symbols. The following table shows the relationship between representation and status:

Representation	Status
Green solid	Satisfied
Blue dashed	Not satisfied
Gray solid	Unknown or not executed
Black	Not interconnected
Parameter in a frame with a saturation of 100 %	Value is current
Parameter in a frame with a saturation of 50 %	Value originates from an earlier cycle. The point in the program was not executed in the current cycle.

The values of the operands are displayed above the relevant operand name in a gray box.

Program status display for SCL programs

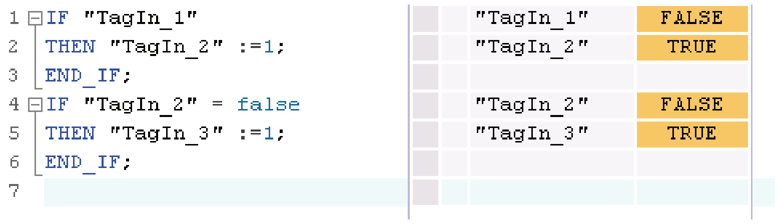
Displays in program status

The display of the program status is updated cyclically and shown in a table. The table is displayed immediately beside the SCL program and you can see the program status for each line of the program. The table contains the following information:

- Tag names
- Value

You can move the table to the left or right at any time.

The following figure shows an example of the program status display for SCL:



```
IF "TagIn_1"
THEN "TagIn_2" :=1;
END_IF;
IF "TagIn_2" = false
THEN "TagIn_3" :=1;
END_IF;

```

"TagIn_1"	FALSE
"TagIn_2"	TRUE
"TagIn_2"	FALSE
"TagIn_3"	TRUE

In the first column, you can see the name of the tag for which the current value is being displayed. If the line includes the "IF", "WHILE" or "REPEAT" instruction, the result of the instruction is displayed in the line as "True" or "False". If the line contains more than one tag, the value of the first tag is displayed. In both cases, all tags of these lines are displayed with their values in a separate list as soon as you select a line. If you place the cursor in a tag in the program code, this is shown in bold face in the list. You can also display the other tags of a line explicitly by clicking the arrow right located in front of lines containing more than one tag.

If the code of the line is not executed, the tag name is displayed in the values table in gray text.

The current values of the tags are displayed in the last column. If no values can be displayed for a tag, the line has a yellow background and three question marks are shown. In this case, select the "Create extended status information" check box in the properties of the block and download the block to the device again. All values are then displayed.

Switching test with program status on/off

You can activate the program status for all programming languages. For the graphic programming languages LAD and FBD, you can also enable the program status at a specific position or for a specific selection.

Requirement

- The identical block exists in the device.
- The block is open.

Switching the program status on or off

To switch the program status for a block on or off, follow these steps:

1. Click the "Monitoring on/off" button in the toolbar.

   If you have not already established an online connection, the "Go online" dialog opens. In this dialog, you can establish an online connection.

   See also: Establishing and terminating an online connection

Enabling program status starting at a specific point in a network

To start the program status for LAD and FBD at a specific point, follow these steps:

1. Click the "Monitoring on/off" button in the toolbar.
2. Right-click on the tag you want program status to start from.
3. Select "Modify > "Monitor from here" in the shortcut menu.

Enabling program status for selected tags

To start the program status for LAD and FBD for selected tags, follow these steps:

1. Click the "Monitoring on/off" button in the toolbar.
2. Select the tags for which you want to start the program status.
3. Select "Modify > Monitor selection" in the shortcut menu.

Note

The resources for testing with program status are limited. If there are not enough resources for the current test, earlier tests will be terminated.

Result

If you enable the display of the program status, an online connection is established and the program status is displayed. When you turn off the display of the program status, you can terminate the online connection at the same time.

If the CPU is in "HOLD" or "STOP" mode, the call hierarchy of the block is displayed in the "Call hierarchy" pane on the "Testing" task card. With S7-1200 CPUs, the call hierarchy is also displayed during the test with program status. Using this call hierarchy, you can change to one of the calling blocks.

Editing blocks during the program test

If you edit blocks while the test with program status is still running, online monitoring will be interrupted and you will be able to edit the block offline. If the block is not available offline in the project, you will first have to load it from the device to the project. After editing the block, you will also have to compile and download it again.

Procedure

To edit blocks while the test with program status is still running, follow these steps:

1. Edit the block as necessary.

   The test with program status is interrupted and the block is switched offline assuming it exists offline.
2. If the block does not exist offline, load it to the project from the device.
3. Compile the block.

   See also: Compiling blocks (Page 1071)
4. Download the block to the device.

   See also: Downloading blocks (Page 1076)

Result

The block now contains your modifications both online and offline. The online connection is re-established and testing with program status continues.

Modifying tags in the program status

While testing with the program status, you have the option of modifying tags to the following values once and immediately:

- Modify to 1

  Modifies tags of the "Bool" data type to the value "True".
- Modify to 0

  Modifies tags of the "Bool" data type to the value "False"
- Modify operand

  You can enter a modify value for tags that do not belong to the "Bool" data type.

Note that you cannot modify peripheral inputs, for example, via TagName:P.

Procedure

To modify tags during testing with the program status, proceed as follows:

1. Right-click on the tag you want to modify.
2. Select one of the following commands in the shortcut menu:
   - "Modify > Modify to 1"
   - "Modify > Modify to 0"
   - "Modify > Modify operand"
3. If you select "Modify operand", the "Modify operand" dialog opens. Enter the value you require in the "Modify value" box and confirm with "OK".

9.4.3 Testing with the watch table

9.4.3.1 Introduction to testing with the watch table

Overview

The following functions are available in the watch table:

- Monitoring tags
  This displays the current values of the individual tags of a user program or a CPU on the programming device or PC.
- Modifying tags
  You can use this function to assign specific values to the individual tags of a user program or CPU. Modifying is also possible with Test with program status .
- "Enable peripheral outputs" and "Modify now"
  These two functions enable you to assign specific values to individual peripheral outputs of a CPU in STOP mode. You can also use them to check your wiring.

Monitoring and modifying tags

The following tags can be monitored and modified:

- Inputs, outputs, and bit memory
- Contents of data blocks
- I/O

Possible applications

The advantage of the watch table is that a variety of test environments can be stored. This enables you to reproduce tests during commissioning or for service and maintenance purposes.

See also

Creating and editing watch tables (Page 1137)

Layout of the watch table (Page 1134)

Basic mode and expanded mode in the watch table (Page 1135)

Icons in the watch table (Page 1136)

9.4.3.2 Layout of the watch table

Introduction

A watch table contains the tags you defined for the entire CPU. A "Watch and force tables" folder is automatically generated for each CPU created in the project. You create a new watch table in this folder by selecting the "Add new Watch table" command.

Layout of the watch table

The columns displayed in the watch table depend on the mode you are working in: basic mode or expanded mode.

The following additional columns are shown in expanded mode:

- Monitor with trigger
- Modify with trigger

The names of the columns can also be changed dynamically based on the action.

Meaning of the columns

The following table shows the meaning of the individual columns in basic mode and expanded mode:

Mode	Column	Meaning
Basic mode	i	Identifier column
	Name	Name of the inserted tag
	Address	Address of the inserted tag
	Display format	Selected display format
	Monitor value	Values of the tags, depending on the selected display format.
	Modify value	Value with which the tag is modified.
	⚡	Select the tag to be modified by clicking the corresponding check box.
	Comment	Comment for documentation of the tags
The following additional columns are shown in expanded mode:	Monitor with trigger	Display of selected monitoring mode
	Modify with trigger	Display of selected modify mode

See also

Icons in the watch table (Page 1136)

9.4.3.3 Basic mode and expanded mode in the watch table

Difference between basic mode and expanded mode in the watch table

Depending on the mode specified, the watch table displays different columns and column headings that can be used to perform different actions.

You will find a detailed list of the columns in Layout of the watch table (Page 1134).

Switching between basic mode and expanded mode

You have the following options of toggling between the basic and expanded mode:

- Click the icon "Show/hide advanced setting columns". Click this icon again to return to the basic mode.

  Or:

- Activate the check box for the "Expanded Mode" command in the "Online" menu. Deactivate this check box to return to the basic mode.

Functionality in expanded mode

The following functionality is only possible in expanded mode:

- Monitor with trigger
- Modify with trigger
- Enable peripheral outputs

See also

Setting the monitoring and modify mode (Page 1145)

9.4.3.4 Icons in the watch table

Meaning of the icons

The following table shows the meaning of the icons in the watch table:

Icon	Meaning
▦	Identifies a table inside the project tree as a watch table.
i	Shows information in the identifier column.
⚡1	Modifies the addresses of all selected tags immediately and once. This command is executed once and as quickly as possible without reference to a defined trigger point in the user program.
⚡🕒	Modifies the addresses of all selected tags with reference to a defined trigger point in the user program.
⚡✓	Disables the command output disable of the peripheral outputs. You can then modify the peripheral outputs when the CPU is in STOP mode.
▥🕒	Displays all columns of expanded mode. If you click this icon again, the columns of expanded mode will be hidden.
▥⚡	Displays all modify columns. If you click this icon again, the modify columns will be hidden.
▷	Starts monitoring of the visible tags in the active watch table. The default setting for the monitoring mode in basic mode is "permanent". In expanded mode, you can set defined trigger points for the monitoring of tags.
1	Starts monitoring of the visible tags in the active watch table. This command is executed immediately and the tags are monitored once.
⚡	Displays the check box for the selection of tags to be modified.
🟩	Indicates that the value of the selected tag has been modified to "1".
▨	Indicates that the value of the selected tag has been modified to "0".

Icon	Meaning
=	Indicates that the address is being used multiple times.
	Indicates that the substitute value is being used. Substitute values are values that are output to the process in case of signal output module faults or are used instead of a process value in the user program in case of signal input module faults. The substitute values can be assigned by the user (e.g., retain old value).
	Indicates that the address is blocked because it is already being modified.
	Indicates that the address cannot be modified.
	Indicates that the address cannot be monitored.
F	Indicates that an address is being forced.
F	Indicates that an address is being partly forced.
F	Indicates that an associated I/O address is being fully or partly forced.
F	Indicates that an address cannot be fully forced. Example: It is indeed possible to force the address QW0:P, but it is not possible to force the address QD0:P since this address area is eventually not available on the CPU.
✖	Indicates that a syntax error occurred.
⚠	Indicates that the address is selected but at the moment e.g. has not yet been modified.

See also

Layout of the watch table (Page 1134)

9.4.3.5 Creating and editing watch tables

Creating a watch table

Introduction

The watch table allows you to monitor and modify tags in the user program. Once you have created a watch table, you can save it, duplicate it, and print it and use it again and again to monitor and modify tags.

Requirement

A project is open.

Procedure

To create a watch table, follow these steps:

1. Click "Project view" in the status bar.

   The project view is displayed.
2. In the project tree, double-click the CPU for which you want to create a watch table.
3. Double-click the "Watch and force tables" folder and then the "Add new watch table" command.

   A new watch table is added.
4. In the "Name" column or in the "Address" column, enter the name or the absolute address for the tags that you want to monitor or modify.
5. You can select a display format from the drop-down list in the "Display format" column if you want to change this default setting.
6. Now decide whether you want to monitor or modify the entered tags and, if applicable, enter the desired values for modifying.

Opening a watch table

Requirement

A watch table has been created.

Procedure

To open a watch table, follow these steps:

1. Open the "Watch and force tables" folder below the desired CPU.
2. Double-click on the required watch table in the folder.

Result

The selected watch table opens.

Copying and pasting a watch table

Requirement

A watch table has been created.

Procedure

To copy a watch table, follow these steps:

1. Right-click the watch table that you want to copy.

2. In the context menu, select "Copy".
3. In the project tree, open the folder structure for the CPU in which you want to paste the copied watch table.
4. Right-click on the "Watch and force tables" folder.
5. In the context menu, select "Paste".
6. Alternatively, you can select the entire contents of the watch table and Drag & Drop it onto another watch table.

Result

A copy of the selected watch table is placed in the "Watch and force tables" folder of the relevant CPU.

Saving a watch table

Prerequisite

A watch table has been created.

Procedure

To save a watch table, follow these steps:

1. In the project tree select the watch table you want to save.
2. If you wish to change the preset name of the table, select the "Rename" command in the context menu and enter a new name for the table.
3. In the "Project" menu, select "Save". Note that this save operation will save the entire project.

Result

The contents of the watch table and the project are saved.

Note

You can reuse saved watch tables to monitor and modify tags when retesting your program.

9.4.3.6 Entering tags in the watch table

Basic information on entering tags in the watch table

Recommended procedure

- Select the tags whose values you want to monitor or modify, and enter them in the watch table.
- When entering tags into the watch table, please note that these tags must be previously defined in the PLC tag table.
- When entering tags, work from the outside to the inside. This means that you start by entering the tags for the inputs in the watch table. Then, you enter the tags that are affected by the inputs or that affect the outputs. Finally, you enter the tags for the outputs.

Example of filling out a watch table

- Enter the absolute address to be monitored or modified in the "Address" column.
- Enter the symbolic name for the tag in the "Name" column.
- Select the display format you require from the drop-down list in the "Display format" column, if you do not want to use the default setting.
- Now decide whether you want to monitor or modify the entered tags. Enter the desired values for modifying as well as a comment in the corresponding columns of the watch table.

Syntax check

When you enter the tags in the watch table, the syntax of each cell is checked when you exit the cell. Incorrect entries are marked in red.

Note

When you place the mouse pointer in a cell marked in red, brief information is displayed with additional notes on the error.

See also

Permitted operands for the watch table (Page 1141)

Permissible modify values for the watch table (Page 1141)

Permitted operands for the watch table

Permissible operands for the watch table

The following table shows the operands that are permitted for the watch table:

Permitted operand	Example of data type	Example (International mnemonics)
Input/output/bit memory	BOOL	I1.0, Q1.7, M10.1 I0.0:P; Q0.0:P
Input/output/bit memory	BYTE	IB1/QB10/MB100 IB1:P; QB1:P
Input/output/bit memory	WORD	IW1; QW10; MW100 IW2:P; QW3:P
Input/output/bit memory	DWORD	ID4; QD10; MD100 ID2:P; QD1:P
Data block	BOOL	DB1.DBX1.0
Data block	BYTE	DB1.DBB1
Data block	WORD	DB1.DBW1
Data block	DWORD	DB1.DBD1

Note

Please observe the following notes to work with the watch table.

- You cannot enter "DB0..." because it is used by the system!
- Peripheral outputs can be modified but not monitored.
- Peripheral inputs can be monitored but not modified.

See also

Basic information on entering tags in the watch table (Page 1140)

Permissible modify values for the watch table

Entry of modify values in the watch table

The following table shows the operands that are permitted for the entry of modify values in the watch table:

Table 9- 14 Bit operands

Possible bit operands	Example for permitted modify values
I1.0	True
M1.7	False
Q1.0	0
Q1.1:P	1
DB1.DBX1.1	2#0
M1.6	2#1

Table 9- 15 Byte operands

Possible byte operands	Example for permitted modify values
IB1	2#00110011
MB12	B#16#1F
QB10	1F
QB11:P	'a'
DB1.DBB1	10

Table 9- 16 Word operands

Possible word operands	Example for permitted modify values
IW1	2#0011001100110011
MW12	W#16#ABCD
MW14	ABCD
QW10	B#(12, 34)
QW12:P	12345
DB1.DBW1	'ab'
MW16	S5T#9s_340ms
MW18	C#123
MW9	D#2006-12-31

Table 9- 17 Double word operands

Possible double word operands	Example for permitted modify values
ID1	2#00110011001100110011001100110011
QD10	Dw#16#abcdef10
QD12:P	ABCDEF10
DB1.DBD2	b#(12,34,56,78)
MD8	L#-12
MD12	L#12
MD16	123456789

Possible double word operands	Example for permitted modify values
MD20	123456789
MD24	T#12s345ms
MD28	Tod#1:2:34.567
MD32	P#e0.0

Overview of the display formats

Display formats in the watch table

The display format you select specifies the representation of a tag value.

When entering the address a display format is automatically preset. If you want to change this, you can select a display format from the drop-down list in the "Display formats" column. The drop-down list only offers the display formats which are valid for this data type. The display format that appears first in the list is the pre-selected format.

Example

The following table shows the data types permitted for the watch table and their possible display formats:

Data type	Possible display formats
BOOL	Bool, Hex, BCD, Octal, Bin, Dec, Dec+/-
BYTE	Hex, BCD, Octal, Bin, Dec, Dec+/-, Character
WORD	Hex, BCD, Octal, Bin, Dec, Dec+/-, Dec_Sequence, Character, SIMATIC_Timer, Date
DWORD	Hex, BCD, Octal, Bin, Dec, Dec+/-, Dec_Sequence, Character, Floating-point number, Time of day, Timer, Pointer
SINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character
INT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, SIMATIC_Timer, Counter
DINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Floating-point number, Time of day, Timer, Pointer
USINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character
UINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, SIMATIC_Timer, Counter
UDINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Floating-point number, Time of day, Timer, Pointer
BCD16	BCD, Hex, Octal, Bin, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Date, SIMATIC_Timer, Counter
BCD32	BCD, Hex, Octal, Bin, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Floating-point number, Time of day, Timer, Pointer

Data type	Possible display formats
REAL	Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Time of day, Timer, Pointer
LREAL	Floating-point number Note: The display of LREAL in the watch table is limited to 13 digits plus exponent.
DATE	Date, Hex, BCD, Bin
TIME_OF_DAY	Time of day, Hex, BCD, Bin
TIME	Timer, Hex, BCD, Bin
DATE_AND_TIME	Date and time,
TIMER	SIMATIC_Timer, Hex, BCD, Bin
CHAR	Character, Hex, BCD, Octal, Bin, Dec, Dec+/-
STRING	Character string
POINTER	Pointer, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Floating-point number, Time of day, Timer, Number
COUNTER	Counter, Hex, BCD, Bin

For more information, refer to the description of the valid data types.

Selecting the display format for tags

Procedure

To select the display format of the tags, follow these steps:

1. Enter the desired address in the watch table.
2. Click the desired cell in the "Display format" column, and open the drop-down list.

   The permissible display formats are shown in the drop-down list.
3. Select the desired display format from the drop-down list.

Note

If the selected display format cannot be applied, the "hexadecimal" display format is automatically shown.

9.4.3.7 Monitoring tags in the watch table

Introduction to monitoring tags in the watch table

Introduction

The watch table allows you to monitor the tags of the configured input and output modules in the CPU, depending on the monitoring and modify mode (Page 1145) selected. To monitor tags, an online connection to the CPU must exist.

Options for monitoring tags

The following options are available for monitoring tags:

- Monitor now
  This command starts the monitoring of the visible tags in the active watch table immediately and once only.
- Monitor all
  This command starts the monitoring of all visible tags in the active watch table, depending on the selected watch mode:
  - In basic mode, the monitoring mode is set to "permanent" by default.
  - In expanded mode, you can specify defined trigger points for the monitoring of tags.

  Note

  If the monitoring mode is changed while in expanded mode and then a switch is made to basic mode, the monitoring mode set before will also be applied in basic mode.

CPU-specific limitations when monitoring tags

The following CPU-specific differences exist:

- CPU S7-300/400:
  CPUs from this family can monitor the first 30 characters of a string.
- CPU S7-1200:
  CPUs from this family can monitor a string up to the total size of 254 characters.

Setting the monitoring and modify mode

Introduction

By selecting the monitoring and modify mode, you specify the trigger point and the duration of the tag monitoring in the watch table and the force table.

Possible monitoring and modify modes (duration of monitoring or modifying)

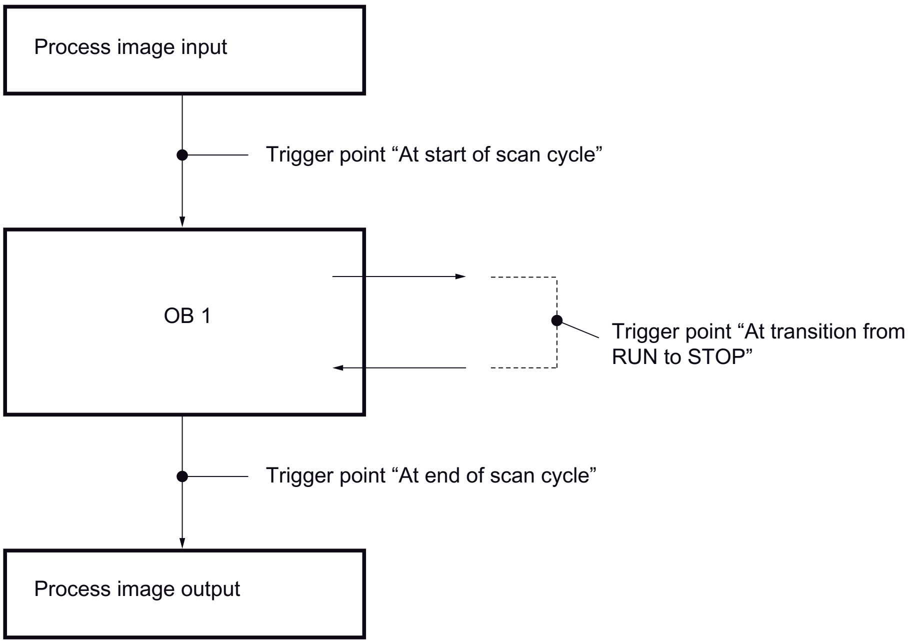
The following monitoring and modify modes are available:

- Permanent
  - In this mode, the inputs can be monitored at the start of the scan cycle and the outputs at the end.
- Once only, at start of scan cycle
- Once only, at end of scan cycle
- Permanently, at start of scan cycle
- Permanently, at end of scan cycle
- Once only, at transition to STOP
- Permanently, at transition to STOP

Selecting the trigger point

The trigger points "Beginning of scan cycle", "End of scan cycle", and "Switch to stop" specify the time at which the tags are to be read from the CPU or updated in the CPU.

The following diagram shows the position of these trigger points:


Process image input
Trigger point “At start of scan cycle”
OB 1
Trigger point “At transition from RUN to STOP”
Trigger point “At end of scan cycle”
Process image output

Position of the trigger points

From the position of the trigger points, it follows that:

- Modifying of inputs is only appropriate at the beginning of the scan cycle (corresponding to the beginning of the user program OB 1), because otherwise the process image input is updated again after modifying and is thus overwritten.
- Modifying of outputs is only appropriate at the end of the scan cycle (corresponding to the end of the user program OB 1), because otherwise the process image output can be overwritten by the user program.
- The modified value is displayed in the "Monitor value" column, provided that monitoring is active and the modified value is not overwritten by the user program.

Monitoring of tags

When tags are being modified, the following applies to the trigger points:

- If you have specified the modify mode as "once only", you will receive a message if the selected tags cannot be modified.
- In "permanent" modify mode, you do not receive a message.

Note regarding the "Modify now" command

You can modify the values of selected tags immediately using the "Online > Modify >Modify now" command. This command is executed once only and as quickly as possible without reference to a defined position (trigger point) in the user program. This function is used mainly for modifying when the CPU is in STOP mode.

"Monitor all" command for tags

Introduction

The "Monitor all" command allows you to start monitoring the visible tags in the active watch table. The default setting for the monitoring mode in basic mode of the watch table is "permanent". In expanded mode, you can specify defined trigger points for the monitoring of tags. In this case, the tags are monitored with reference to the specified trigger points.

Prerequisite

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To execute the "Monitor all" command, follow these steps:

1. Enter the tags to be monitored and the corresponding addresses in the watch table.

2. Switch to expanded mode by clicking

   the icon "Show/hide advanced setting columns" in the toolbar.
3. If you want to change the default monitoring mode for a tag, click the appropriate cell in the "Monitor with trigger" column and select the desired monitoring mode from the drop-down list.
4. Click the "Monitor all" icon in the toolbar.

Result

The tags of the active watch table are monitored using the monitoring mode selected.

See also

Icons in the watch table (Page 1136)

Entering tags in the watch table (Page 1140)

Basic mode and expanded mode in the watch table (Page 1135)

"Monitor now" command for tags

Introduction

The "Monitor now" command starts the monitoring of tags immediately without reference to defined trigger points. The tag values are read out once only and displayed in the watch table.

Requirement

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To execute the "Monitor now" command, follow these steps:

1. Enter the tags to be monitored and the corresponding addresses in the watch table.
2. Click the "Monitor now" icon in the toolbar.

Result

The tags of the active watch table are monitored immediately and once only.

See also

Icons in the watch table (Page 1136)

Entering tags in the watch table (Page 1140)

Basic mode and expanded mode in the watch table (Page 1135)

9.4.3.8 Modifying tags in the watch table

Introduction to modifying tags

Introduction

The watch table allows you to modify the tags of the configured input and output modules in the CPU, depending on the monitoring and modify mode (Page 1145) selected. To monitor the tags, an online connection to the CPU must exist.

> ⚠ DANGER
>
> **Danger when modifying:**
>
> Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
> Make certain that dangerous conditions cannot occur before you execute the "Modify" function.

Options for modifying tags

The following options are available for modifying tags:

- Modify to "0"

  This command modifies the selected address to the modify value "0".
- Modify to "1"

  This command modifies the selected address to the modify value "1".
- Modify once only and immediately

  This command modifies all selected addresses in the active watch table "once only and immediately".

- Modify with trigger

  This command modifies all selected addresses in the active watch table using the monitoring and modify mode (Page 1145) selected.

  The "Modify with trigger" function is only available in expanded mode. You will not receive a message indicating whether or not the selected addresses were actually modified with the specified value. You should use the "Modify once only and immediately" function if you require such a confirmation.

- Enable peripheral outputs

  This command disables the command output disable.

  This function can only be executed in expanded mode, when the CPU is in STOP and the option Force (Page 1170) of tags is not enabled. If desired, deactivate this function in the force table.

Note

When modifying, note the following:

Modifying can **not** be undone.

Modify tags to "0"

Introduction

You can assign one-time values to tags independent of the monitoring and modify mode and modify them. The modify command is executed as fast as possible, similar to a "Trigger now" command, without reference to a defined position in the user program.

⚠ DANGER

Danger when modifying:

Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
Make certain that dangerous conditions cannot occur before you execute the "Modify" function.

Prerequisite

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To modify tags to "0", follow these steps:

1. Enter the desired address in the watch table.

2. Select the "Online > Modify > Modify to 0" command in order to modify the selected address with the specified value.

Result

The selected address is modified to "0".

Note

When modifying, note the following:

Modifying can **not** be undone!

Modify tags to "1"

Introduction

You can assign one-time values to tags independent of the monitoring and modify mode and modify them. The modify command is executed as fast as possible, similar to a "Trigger now" command, without reference to a defined position in the user program.

DANGER

Danger when modifying:

Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
Make certain that dangerous conditions cannot occur before you execute the "Modify" function.

Prerequisite

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To modify tags to "1", follow these steps:

1. Enter the desired address in the watch table.
2. Select the "Online > Modify > Modify to 1" command in order to modify the selected address with the specified value.

Result

The selected address is modified to "1".

Note

When modifying, note the following:

Modifying can **not** be undone!

"Modify now" command for tags

Introduction

You can assign one-time values to tags independent of the monitoring and modify mode and modify them immediately. The modify command is executed as fast as possible, similar to a "Trigger now" command, without reference to a defined position in the user program.

⚠ DANGER

Danger when modifying:

Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
Make certain that dangerous conditions cannot occur before you execute the "Modify" function.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To modify tags immediately, follow these steps:

1. Enter the desired addresses and modify values in the watch table.
2. Select the addresses to be modified by selecting the check boxes for modifying in the column after the "Modify value".
   A yellow triangle appears behind the selected check box, indicating that the address is now selected for modifying but has not yet been modified.
3. Select the "Online > Modify > Modify once and now" command in order to immediately modify the selected address once only with the specified value.

Result

The selected addresses are modified immediately and once only.

Note

When modifying, note the following:

Modifying can **not** be undone!

"Modify with trigger" command for tags

Introduction

You can assign values to addresses dependent on the defined monitoring and modify mode and modify them. The modify command is executed as specified in the monitoring and modify mode, with reference to the defined trigger position in the user program.

⚠ DANGER

Danger when modifying:

Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
Make certain that dangerous conditions cannot occur before you execute the "Modify" function.

Prerequisites

- A watch table has been created.
- An online connection to the CPU exists.
- The watch table has to be in expanded mode.

Procedure

To modify tags "with trigger", follow these steps:

1. Enter the desired addresses and modify values in the watch table.
2. Select the addresses to be modified by selecting the check boxes for modifying in the column after the "Modify value".
   A yellow triangle appears behind the selected check box, indicating that the address is now selected for modifying but has not yet been modified.
3. Switch to expanded mode using the icon "Show/hide advanced settings columns" in the toolbar or the "Online > Expanded mode" command.

   The "Monitor with trigger" and "Modify with trigger" columns are displayed.

4. In the "Modify with trigger" column, select the desired modify mode from the drop-down list box. The following options are available:
   - Permanent
   - Permanently, at start of scan cycle
   - Once only, at start of scan cycle
   - Permanently, at end of scan cycle
   - Once only, at end of scan cycle
   - Permanently, at transition to STOP
   - Once only, at transition to STOP
5. Start modifying using the "Online > Modify > Modify with trigger" command.
6. Confirm the prompt with "Yes" if you want to start modifying with trigger.

Result

The selected tags are modified using the selected monitoring and modify mode. The yellow triangle is no longer displayed.

Note

When modifying, note the following:

Modifying can **not** be undone!

Enable peripheral outputs

Introduction

The "Enable peripheral outputs" function deactivates the command output disable of the peripheral outputs. You can then modify the peripheral outputs when the CPU is in STOP mode. This function is available in the watch table in "Expanded mode" only.

⚠ DANGER

Danger when enabling the peripheral outputs:

Attention, the enabling of the peripheral outputs can cause serious personal injury and material damage!
Make certain that dangerous conditions cannot occur before you execute the "Enable peripheral outputs" function.

Prerequisites

- A watch table has been created.
- An online connection to the CPU exists.
- The CPU is in STOP mode before you can enable the peripheral outputs.
- The watch table has to be in expanded mode.
- The option Force (Page 1170) of tags must not be enabled.

Note

"Enable peripheral outputs" function

- This function is possible only in STOP mode. The function is exited by an operating state change of the CPU and by the termination of the online connection.
- While the function is enabled, forcing is not possible.

Procedure

To enable the peripheral outputs in STOP mode, follow these steps:

1. Enter the desired addresses and modify values in the watch table.
2. Select the addresses to be modified by selecting the check boxes for modifying in the column after the "Modify value".
   A yellow triangle appears behind the selected check box, indicating that the address is now selected for modifying but has not yet been modified.
3. Switch to expanded mode using the icon "Show/hide advanced settings columns" in the toolbar or the "Online > Expanded mode" command.

   The "Monitor with trigger" and "Modify with trigger" columns are displayed.
4. Change the relevant CPU to STOP using the operator panel.
5. Right-click to open the shortcut menu and select "Enable peripheral outputs".
6. Confirm the prompt with "Yes" if you want to unlock the command output disable for the peripheral outputs.
7. Modify the peripheral outputs using the "Online > Modify > Modify now" command.

Result

The peripheral outputs are modified with the selected modify values. The yellow triangle is no longer displayed.

Enabling the peripheral outputs

The "Enable peripheral outputs" function remains active until:

- The "Enable peripheral outputs" command is deactivated again via the shortcut menu or via the "Online > Modify > Enable peripheral outputs" command.
- The CPU is no longer in STOP mode.
- The online connection is terminated.

Note

When modifying, note the following:

Modifying can **not** be undone!

9.4.4 Testing with the force table

9.4.4.1 Introduction for testing with the force table

Overview

You can use the force table to assign permanent values to individual tags

of the user program. This action is referred to as "forcing".

The following functions are available in the force table:

- **Monitoring tags**
  This displays the current values of the individual tags of a user program or a CPU on the programming device or PC. Tags can be monitored with or without a trigger condition.
- **Forcing tags**
  This function lets you assign a permanent value to individual peripheral tags of the user program.

Monitoring and forcing tags

The monitoring and forcing of tags is always dependent on the operand scope of the CPU used.

The following tags can be monitored:

- Inputs, outputs, and bit memories
- Contents of data blocks
- Peripheral inputs

The following tags can be forced:

- Peripheral inputs
- Peripheral outputs

Example

- Independent of the CPU used, only I/O can be forced, such as: "Tag_1":P or "QW0:P" or "IW0:P". Note that "Tag_1":P must not be the symbolic name of a bit memory.

Possible applications

One advantage of the force table is that you can simulate different test environments and overwrite tags in the CPU with a permanent value. This enables you to intervene in the ongoing process for regulating purposes.

See also

Layout of the force table (Page 1159)

Basic mode and expanded mode in the force table (Page 1160)

Icons in the force table (Page 1160)

Open and edit force table (Page 1161)

9.4.4.2 Safety precautions when forcing tags

Safety precautions when forcing tags

Because the forcing function allows you to intervene permanently in the process, observance of the following notices is essential:

⚠ DANGER

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

⚠ CAUTION

Prevent personal injury and material damage!

- Before you start the "Force" function, you should ensure that no one else is currently executing this function on the same CPU.
- Forcing can only be stopped by clicking the "Stop forcing" icon or using the "Online > Force > Stop forcing" command. Closing the active force table does **not** stop the forcing!
- Forcing can **not** be undone!
- Review the differences between " modifying tags" (Page 1149) and "forcing tags" (Page 1170).
- If a CPU does not support the "Force" function, the relevant icons cannot be selected.
- If the function "Enable peripheral outputs" is active on your CPU, then forcing is **not** possible on this CPU. If desired, deactivate this function in the watch table.

9.4.4.3 Layout of the force table

Introduction

In the force table, enter the CPU-wide tags that you have defined and selected and which are to be forced in the allocated CPU. Only peripheral inputs and peripheral outputs can be forced.

For each CPU created in the project, a force table will automatically be created in the "Watch and force tables" folder. Only one force table can be allocated to a CPU. This force table displays all the addresses forced in the allocated CPU.

Layout of the force table

The columns displayed in the force table depend on the mode you are working in: basic mode or expanded mode.

In expanded mode the "Monitor with trigger" column is also displayed.

Meaning of the columns

The following table shows the meaning of the individual columns in basic mode and expanded mode:

Mode	Column	Meaning
Basic mode	i	Identification column
	Name	Name of the inserted tag
	Address	Address of the inserted tag
	Display format	Selected display format
	Monitor value	Values of the tags, dependent on the selected display format.
	Force value	Value with which the tag is forced.
	F ("Force")	Select the tag to be forced by activating the corresponding check box.
	Comment	Comment for documentation of the tags
The following additional column is shown in expanded mode:	Monitor with trigger	Display of selected monitoring mode

See also

Icons in the force table (Page 1160)

Basic mode and expanded mode in the force table (Page 1160)

9.4.4.4 Basic mode and expanded mode in the force table

Difference between basic mode and expanded mode in the force table

In expanded mode the "Monitor with trigger" column is also displayed in the force table.

You will find a detailed list of the columns under Layout of the force table (Page 1159).

Switching between basic mode and expanded mode

You have the following options of toggling between the basic and expanded mode:

- Click the icon "Show/hide advanced setting columns". Click this icon again to return to the basic mode.

  Or:

- Activate the check box for the "Expanded mode" command in the "Online" menu. Deactivate this check box to return to the basic mode.

Functionality in expanded mode

The "Monitor with trigger" function can only be selected in expanded mode.

9.4.4.5 Icons in the force table

Meaning of the icons

The following table shows the meaning of the icons in the force table:

Icon	Meaning
	Identifies a table inside the project tree as a force table.
i	Identification column
	Displays all columns of expanded mode. If you click this icon again, the columns of the expanded mode will be hidden.
F▷	Starts forcing for all addresses of the selected tags. If forcing is already running, the previous action is replaced without interruption.
F■	Stops forcing of addresses in the force table.
▷	Starts monitoring of the visible tags in the force table. The default setting for monitoring in basic mode is "permanent". In expanded mode an additional column is shown and you can set certain trigger points for monitoring tags.
1	Starts monitoring of the visible tags in the force table. This command is executed immediately and the tags are monitored once.
F	Displays the check box for the selection of tags to be forced.

Icon	Meaning
F⊘	Indicates that an address cannot be forced.
F⊘	Indicates that an address cannot be fully forced. Example: It is possible to force the address QW0:P, but it is not possible to force the address QD0:P because this address area is potentially not available on the CPU.
⊘	Indicates that an address cannot be monitored.
F	Indicates that an address is being forced.
F	Indicates that an address is being partly forced.
F	Indicates that the associated peripheral address is being forced.
✖	Indicates that a syntax error occurred.
⚠	Indicates that the address is selected but has not been forced yet.

See also

Layout of the force table (Page 1159)

9.4.4.6 Open and edit force table

Display force table

Introduction

You cannot create a new force table; one force table already exists for each CPU. It is permanently allocated to this CPU and cannot be copied or duplicated.

Requirements

A project with an allocated CPU has to be open.

Displaying a force table

The force table is always displayed below a CPU in the "Watch and force tables" folder.

Open force table

Requirements

A project with an allocated CPU must be created.

Procedure

Proceed as follows to open a force table:

1. Open the "Watch and force tables" folder below the desired CPU.
2. Double-click the "Force table" in this folder.

Result

The selected force table opens.

Save force table

Requirements

A project with an allocated CPU has been created.

Procedure

Proceed as follows to save a force table:

1. Enter the desired changes in the force table.
2. Select the "Save" command in the "Project" menu or click the "Save project" icon in the toolbar. Note that this save operation will save the entire project.

Result

The contents of the force table and the associated project are saved.

Note

You cannot rename a force table.

9.4.4.7 Entering tags in the force table

Basic principles for entering tags in the force table

Recommended procedure

Select the tags whose values you want to monitor or force, and enter them in the force table.

When entering tags in the force table, please note that these tags must be previously defined in the PLC tag table.

Example of filling out a force table

- You can enter the absolute address that is to be forced or monitored in the "Address" column or you can enter the symbolic name of the tag in the "Name" column.
- Select the display format you require from the drop-down list in the "Display format" column, if you do not want to use the default setting.
- Now you have to decide whether you want to monitor or force the entered tags. Enter the required force value and a comment in the appropriate columns of the force table.
- Note that only peripheral inputs and peripheral outputs can be forced and review the Safety precautions when forcing tags (Page 1173).

Syntax check

When you enter tags in the force table, the syntax of each cell will be checked when you exit the cell. Incorrect entries are marked in red.

Note

When you place the mouse pointer in a cell marked in red, brief information is displayed with additional notes on the error.

Permitted operands for the force table

Permitted operands for the force table

The following table shows the operands that are permitted for forcing in the force table:

Permitted operand	Example of data type	Example (International mnemonics)
Peripheral input/peripheral output	BOOL	I0.0:P; Q0.0:P
Peripheral input/peripheral output	BYTE	IB1:P; QB1:P
Peripheral input/peripheral output	WORD	IW2:P; QW3:P
Peripheral input/Peripheral output	DWORD	ID2:P; QD1:P

The following table shows the operands that are permitted for monitoring in the force table:

Permitted operand	Example of data type	Example (International mnemonics)
Input/output/bit memory	BOOL	I1.0, Q1.7, M10.1 I0.0:P
Input/output/bit memory	BYTE	IB1/QB10/MB100 IB1:P

Permitted operand	Example of data type	Example (International mnemonics)
Input/output/bit memory	WORD	IW1; QW10; MW100 IW2:P
Input/output/bit memory	DWORD	ID4; QD10; MD100 ID2:P
Data block	BOOL	DB1.DBX1.0
Data block	BYTE	DB1.DBB1
Data block	WORD	DB1.DBW1
Data block	DWORD	DB1.DBD1

Note

You cannot enter "DB0..." because it is used by the system!

Permitted force values for the force table

Entering force values in the force table

The following table shows the operands that are permitted for entering force values in the force table:

Table 9- 18 Bit operands

Possible bit operands	Example for permitted force values
I1.0:P	True
I1.1:P	False
Q1.0P	0
Q1.1:P	1
I2.0:P	2#0
I2.1:P	2#1

Table 9- 19 Byte operands

Possible byte operands	Example for permitted force values
IB1:P	2#00110011
IB2:P	B#16#1F
QB14:P	1F
QB10:P	'a'
IB3:P	10

Table 9- 20 Word operands

Possible word operands	Example for permitted force values
IW0:P	2#0011001100110011
IW2:P	W#16#ABCD
QW10:P	ABCD
QW12:P	B#(12, 34)
IW4:P	'ab'
IW6:P	12345
IW8:P	S5T#9S_340ms
IW10:P	C#123
IW12:P	D#2006-12-31

Table 9- 21 Double word operands

Possible double word operands	Example for permitted force values
ID0:P	2#00110011001100110011001100110011
ID4:P	1.2
QD10:P	1.234.e4
QD14:P	Dw#16#abcdef10
ID8:P	16#ABCDEF10
ID12:P	b#(12,34,56,78)
ID16:P	L#-12
ID20:P	L#12
ID24:P	123456789
ID28:P	123456789
ID32:P	T#12s345ms
ID36:P	Tod#14:20:40.645
ID40:P	P#e0.0

Overview of the display formats

Display formats in the force table

The display format you select specifies the representation of a tag value.

When entering the address a display format is automatically preset. If you want to change this, you can select a display format from the drop-down list in the "Display formats" column. The drop-down list only offers the display formats which are valid for this data type. The display format that appears first in the list is the pre-selected format.

Example

The following table shows the data types permitted for the force table and their possible display formats:

Data type	Possible display formats
BOOL	Bool, Hex, BCD, Octal, Bin, Dec, Dec+/-
BYTE	Hex, BCD, Octal, Bin, Dec, Dec+/-, Character
WORD	Hex, BCD, Octal, Bin, Dec, Dec+/-, Dec_Sequence, Character, SIMATIC_Timer, Date
DWORD	Hex, BCD, Octal, Bin, Dec, Dec+/-, Dec_Sequence, Character, Floating-point number, Time of day, Timer, Pointer
SINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character
INT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, SIMATIC_Timer, Counter
DINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Floating-point number, Time of day, Timer, Pointer
USINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character
UINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, SIMATIC_Timer, Counter
UDINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Floating-point number, Time of day, Timer, Pointer
BCD16	BCD, Hex, Octal, Bin, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Date, SIMATIC_Timer, Counter
BCD32	BCD, Hex, Octal, Bin, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Floating-point number, Time of day, Timer, Pointer
REAL	Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Time of day, Timer, Pointer
LREAL	Floating-point number
DATE	Date, Hex, BCD, Bin
TIME_OF_DAY	Time of day, Hex, BCD, Bin
TIME	Timer, Hex, BCD, Bin
DATE_AND_TIME	Date and time,
TIMER	SIMATIC_Timer, Hex, BCD, Bin
CHAR	Character, Hex, BCD, Octal, Bin, Dec, Dec+/-
STRING	Character string
POINTER	Pointer, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Floating-point number, Time of day, Timer, Number
COUNTER	Counter, Hex, BCD, Bin

For more information, refer to the description of the valid data types.

Selecting the display format for tags

Procedure

To select the display format of the tags, follow these steps:

1. Enter the desired address in the force table.
2. Click the desired cell in the "Display format" column, and open the drop-down list.

   The permitted display formats are shown in the drop-down list.
3. Select the desired display format from the drop-down list.

Note

If the selected display format cannot be applied, then the last selected display format will be displayed automatically.

9.4.4.8 Monitoring tags in the force table

Introduction to monitoring tags in the force table

Introduction

Use the force table to monitor the tags of the configured input and output modules in the CPU, dependent on the monitoring mode (Page 1168) you have selected. An online connection to the CPU must exist to monitor tags.

Options for monitoring tags

The following options are available for monitoring tags:

- Monitor all
  This command starts the monitoring of all visible tags in the active force table, dependent on the selected monitoring mode:
  - In basic mode, the monitoring mode is set to "permanent" by default.
  - In expanded mode, you can specify defined trigger points for the monitoring of tags.

  Note

  If the monitoring mode is changed while in expanded mode and then a switch is made to basic mode, the monitoring mode set before will also be applied in basic mode.

- Monitor now
  This command starts the monitoring of the visible tags in the active force table immediately and once only.

CPU-specific limitations when monitoring tags

The following CPU-specific differences exist:

- CPU S7-300/400:
  CPUs from this family can only monitor the first 30 characters of a string.
- CPU S7-1200:
  CPUs from this family can monitor a string up to the total size of 254 characters.

Setting the monitoring mode in the force table

Introduction

By selecting the monitoring mode, you specify the trigger point and the duration of tag monitoring in the force table.

Possible monitoring mode (duration of monitoring)

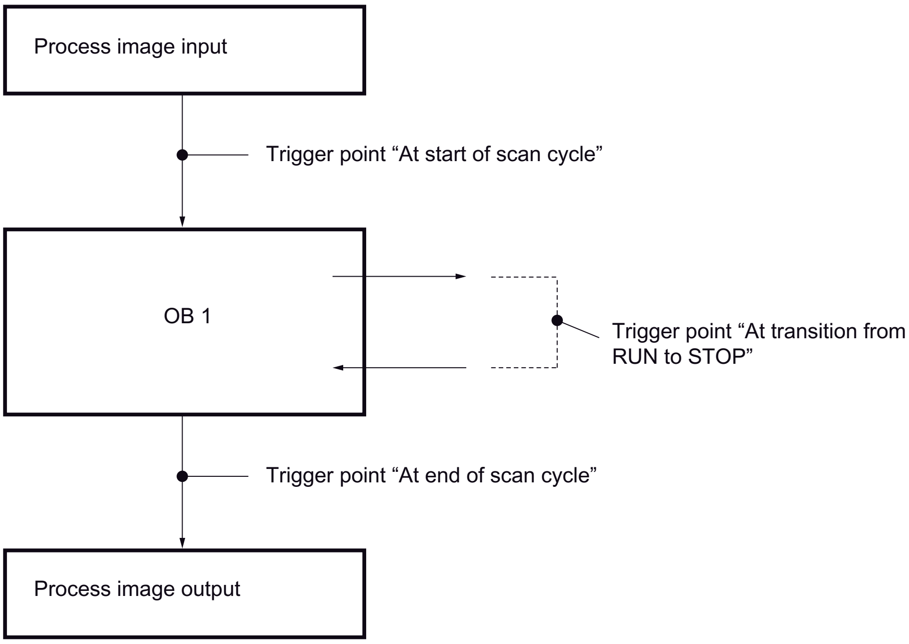
The following selection options are available:

- Permanent: In this mode, the inputs can be monitored at the start of the scan cycle and the outputs at the end.
- Once only, at start of scan cycle
- Once only, at end of scan cycle
- Permanently, at start of scan cycle
- Permanently, at end of scan cycle
- Once only, at transition to STOP
- Permanently, at transition to STOP

Selecting the trigger point

The trigger points "Beginning of scan cycle", "End of scan cycle", and "Switch to stop" specify the time at which the tags are to be read from the CPU or updated in the CPU.

The following diagram shows the position of these trigger points:


Process image input
Trigger point “At start of scan cycle”
OB 1
Trigger point “At transition from RUN to STOP”
Trigger point “At end of scan cycle”
Process image output

"Monitor all" command for tags

Introduction

Use the "Monitor all" command to start monitoring the visible tags in the active force table. In basic mode of the force table, the default setting for the monitoring mode is "permanent". In expanded mode, you can specify defined trigger points for the monitoring of tags. In this case, the tags are monitored with reference to the specified trigger points.

Requirements

- A force table has been created.
- An online connection to the CPU exists.

Procedure

To execute the "Monitor all" command, follow these steps:

1. Enter the tags to be monitored and the corresponding addresses in the force table.

2. Switch to expanded mode by clicking

   the icon "Show/hide advanced setting columns" in the toolbar.
3. If you want to change the default monitoring mode for a tag, click the appropriate cell in the "Monitor with trigger" column and select the desired monitoring mode from the drop-down list.
4. Click the "Monitor all" icon in the toolbar.

Result

The tags of the active force table will be monitored using the set monitoring mode.

"Monitor now" command for tags

Introduction

The "Monitor now" command starts the monitoring of tags immediately without reference to defined trigger points. The tag values are read out only once and displayed in the force table.

Requirements

- A force table has been created.
- An online connection to the CPU exists.

Procedure

To execute the "Monitor now" command, follow these steps:

1. Enter the tags to be monitored and the corresponding addresses in the force table.
2. Click the "Monitor now" icon in the toolbar.

Result

The tags of the active force table are monitored immediately and once only.

9.4.4.9 Forcing tags in the force table

Introduction to forcing tags

Introduction

You can use the force table to assign permanent values to individual tags of the user program. This action is referred to as forcing. Only peripheral inputs and peripheral outputs can be forced.

To use the forcing function, you must have an online connection to the CPU and the utilized CPU must support this functionality.

If you open a force table in the "Watch and force tables" folder below a CPU, all values forced in the allocated CPU will be displayed in this force table, provided that an online connection to the CPU exists.

Possible applications

By permanently assigning defined values to tags, you can specify defined default settings for your user program and, thus, test the programmed functions. Forcing is possible in basic mode and in expanded mode (Page 1160).

Caution when forcing tags

Before forcing, you must review the safety precautions (Page 1173) for this procedure.

⚠ DANGER

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

Options for forcing tags

The following options are available for forcing tags:

- Force to "0"

  This command forces the selected address in the CPU to the force value "0".

- Force to "1"

  This command forces the selected address in the CPU to the force value "1".

- Force all

  This command starts the forcing of enabled addresses in the active force table or replaces an existing force job without interruption.

- Stop forcing

  This command stops the forcing of all addresses in the active force table.

Constraints when forcing tags

Note the following constraints when forcing:

- Forcing is always dependent on the operand scope of the CPU used.
- In principle, only peripheral inputs and peripheral outputs can be forced.
- If the function "Enable peripheral outputs" is active on your CPU, then forcing is not possible. If desired, deactivate this function in the watch table.

Unique aspects when forcing tags

Note that forcing of tags will overwrite values in the CPU and will continue even after the online connection to the CPU is terminated.

- Stop forcing

  Terminating the online connection is not sufficient to stop the forcing operation! To stop forcing, you must select the "Online > Force > Stop forcing" command. Only then will the tags that are visible in the active force table no longer be forced.

- Stop forcing of individual tags

  The "Online > Force > Stop forcing" command always applies to all tags displayed in the force table. To stop forcing individual tags, you must clear the check mark for forcing of these tags in the force table and restart forcing using the "Online > Force > Force all" command.

Safety precautions when forcing tags

Safety precautions when forcing tags

Because the forcing function allows you to intervene permanently in the process, observance of the following notices is essential:

⚠ DANGER

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

⚠ CAUTION

Prevent personal injury and material damage!

- Before you start the "Force" function, you should ensure that no one else is currently executing this function on the same CPU.
- Forcing can only be stopped by clicking the "Stop forcing" icon or using the "Online > Force > Stop forcing" command. Closing the active force table does **not** stop the forcing!
- Forcing can **not** be undone!
- Review the differences between " modifying tags" (Page 1149) and "forcing tags" (Page 1170).
- If a CPU does not support the "Force" function, the relevant icons cannot be selected.
- If the function "Enable peripheral outputs" is active on your CPU, then forcing is **not** possible on this CPU. If desired, deactivate this function in the watch table.

Force tags to "0"

Introduction

You can use the force function to assign permanent values to individual tags of a user program.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 1173).

> ⚠ DANGER
>
> **Prevent personal injury and material damage!**
>
> Note that an incorrect action when executing the "Force" function can:
>
> - Harm persons or pose a health hazard.
> - Cause damage to machinery or the entire plant.

Requirements

- A force table has been created.
- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is **not** enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

To force tags to "0", follow these steps:

1. Open the force table.
2. Enter the desired address in the force table.
3. Select the "Online > Force> Force to 0" command in order to force the selected address with the specified value.
4. Confirm the next dialog with "Yes".

Result

The selected address is forced to "0". The yellow triangle is no longer displayed. A red "F" is displayed in the first column, for example, indicating that the tag is being forced.

Stop forcing

To stop forcing, follow these steps:

1. Open the force table.
2. Select the "Online > Force > Stop forcing" command.
3. Confirm the next dialog with "Yes".

Result

Forcing of the selected values is stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is selected for forcing but is not being forced at the moment.

Note

When forcing, note the following:

- Forcing can **not** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

Force tags to "1"

Introduction

You can use the force function to assign permanent values to individual tags of a user program.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 1173).

⚠ DANGER

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Endanger the life or health of personnel
- Cause damage to machinery or the entire plant.

Requirements

- A force table has been created.
- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is **not** enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

To force tags to "1", follow these steps:

1. Open the force table.
2. Enter the desired address in the force table.
3. Select the "Online > Force> Force to 1" command in order to force the selected address with the specified value.
4. Confirm the next dialog with "Yes".

Result

The selected address is forced to "1". The yellow triangle is no longer displayed. A red "F" is displayed in the first column, for example, indicating that the tag is being forced.

Stop forcing

To stop forcing, follow these steps:

1. Open the force table.
2. Select the "Online > Force > Stop forcing" command.
3. Confirm the next dialog with "Yes".

Result

Forcing of the selected values is stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is selected for forcing but is not being forced at the moment.

Note

When forcing, note the following:

- Forcing can **not** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

"Force all" command for tags

Introduction

You can use the force function to assign permanent values to individual tags of a user program.

If forcing is already active, this forcing operation is replaced without interruption by the "Online > Force > Force all" command. Any forced addresses that are not selected will no longer be forced.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 1173).

⚠ DANGER

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

Requirements

- A force table has been created.
- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is **not** enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

To force tags with the "Online > Force > Force all" command, follow these steps:

1. Open the force table.
2. Enter the desired addresses and force values in the force table.
3. Select the addresses to be forced by selecting the check boxes for forcing in the column after the "Force value".
   A yellow triangle appears behind the selected check box, indicating that the address is selected for forcing but is not being forced at the moment.
4. Select the "Online > Force> Force all" command in order to force the selected addresses with the specified values.
5. Confirm the next dialog with "Yes".

Result

The selected addresses are forced to the specified values. The yellow triangle is no longer displayed. A red "F" is displayed in the first column, for example, indicating that the tag is being forced.

Stop forcing

To stop forcing, follow these steps:

1. Open the force table.

2. Select the "Online > Force > Stop forcing" command.
3. Confirm the next dialog with "Yes".

Result

Forcing of the selected addresses is stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is selected for forcing but is not being forced at the moment.

Note

When forcing, note the following:

- Forcing can **not** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

9.4.4.10 Stop forcing tags

Stop forcing all tags

Introduction

Note the following before you stop forcing tags:

- The stopping of forcing can **not** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 1173).

⚠ DANGER

Prevent personal injury and material damage!

Note that an incorrect action when stopping the "Force" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

Requirements

- A force table has been created in which tags are being forced.
- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is not enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

Proceed as follows to stop **forcing all tags** :

1. Open the force table.
2. Select the "Online > Force > Stop forcing" command in order to stop forcing the displayed addresses.
3. Confirm the "Stop forcing" dialog with "Yes".

Result

The forcing of all tags is stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is flagged for forcing but is not being forced at the moment.

Stop forcing individual tags

Introduction

Note the following before you stop forcing tags:

- The stopping of forcing can **not** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 1158).

⚠ DANGER

Prevent personal injury and material damage!

Note that an incorrect action when stopping the "Force" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

Requirements

- A force table has been created in which tags are being forced.
- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is not enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

Proceed as follows to stop **forcing individual tags** :

1. Open the force table.
2. Deactivate the check boxes for the addresses that are no longer to be forced.
3. Reselect the "Online > Force" command.

Result

Forcing of the disabled addresses will be stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is flagged for forcing but is not being forced at the moment.

9.5 Configuring alarms

9.5.1 Introduction to alarm configuration

Overview

Alarms allow you to detect errors in process control in the automation system quickly, to localize them precisely, and to eliminate them. This leads to a significant reduction in down times in the plant.

Before alarms can be output, they need to be configured.

You can create, edit and compile event-dependent alarms along with their alarm texts and alarm attributes and display them on display devices.

The table below lists the alarm types along with a brief description of their functions.

Alarm type	Description
PLC alarms	PLC alarms are used to report program-synchronous event and are each assigned a block. They are created in the program editor and edited in the alarm editor.
User diagnostic alarms	Using user diagnostic alarms you can write a user entry in the diagnostics buffer and send an appropriate alarm. They are assigned a CPU. They are created in the alarm editor and be edited there.
System alarms	System alarms are configuration-dependent module events and are activated or deactivated in the hardware configuration. They can only be viewed in the alarm editor, but not edited.

9.5.2 Assigning alarm numbers

Assigning numbers

The alarms are identified by a number that is unique within the CPU. This means that it is not necessary to recompile after copying complete programs. Copying a single block is an exception to this rule. Here, recompilation is necessary to include the changed alarm number in the program.

9.5.3 Components of an alarm

Overview

How an alarm is displayed depends on the alarm method, the alarm block used and the display device.

The possible components of an alarm are listed in the following table:

Component	Description
Time stamp	Generated when the alarm event occurs in the automation system.
Alarm state	The following are possible: Incoming, outgoing, outgoing without acknowledgment, outgoing with acknowledgment.
Associated value	With some alarms, it is possible to add a process value that can be evaluated by the alarm instruction being used.
Image	If there is a system crash, the resulting alarms can be displayed later on HMI devices.
Alarm number	A number assigned by the system that is unique within the CPU and identifies an alarm.
Alarm texts	Are configured by the user.

9.5.4 Available alarm blocks

Overview of the alarm blocks

You can select among the following alarm blocks:

- ALARM
- ALARM_8
- ALARM_8P
- NOTIFY
- ALARM_S
- ALARM_SQ
- AR_SEND (for sending archives; no configuration of alarm texts and alarm attributes possible)
- NOTIFY_8P
- ALARM_DQ
- ALARM_D

When do I use which alarm block?

The following table will help you to decide which alarm block to use for your application. The choice of alarm block depends on the following factors:

- The number of channels available in the block and therefore the number of signals to be monitored per block call.
- The possibility of acknowledging alarms.
- The possibility of including associated values.
- The display devices to be used.
- The configuration limits of the CPU.

Alarm block	Channels	Acknowl-edgement	Associated values	Special features
ALARM	1	possible	up to 10	Sends an alarm on rising or falling edge
ALARM_8	8	possible	no	Sends an alarm on rising or falling edge of one or more signals
ALARM_8P	8	possible	up to 10	as ALARM_8
NOTIFY	1	no	up to 10	as ALARM
NOTIFY_8P	8	no	up to 10	as NOTIFY
AR_SEND	1	-	-	Used to send an archive; no configuration of alarm texts and alarm attributes possible
ALARM_SQ	1	possible	1	With each block call and a signal change compared with the previous block call, an alarm is generated
ALARM_S	1	no	1	as ALARM_SQ
ALARM_DQ	1	possible	1	as ALARM_SQ
ALARM_D	1	no	1	as ALARM_SQ

9.5.5 Alarm type and alarms

Using the program and alarm editors, you have the option of creating either an alarm type (i.e. an FB as template for instance DBs) or alarms themselves (i.e. instance DBs as template of an alarm type).

The block with alarm capability can be an FB or an instance DB

- With an FB you create an alarm type that serves as a template for alarms. All the input you make for the alarm type is automatically included for alarms derived from it. If you assign an instance DB to the FB, messages are generated automatically for the instance DB according to the template of the alarm type and alarm numbers are assigned.
- With an instance DB, you can modify the alarms generated based on the alarm type for the specific instance.

The visible difference is that alarm numbers are assigned to alarms whereas they are not assigned to an alarm type. The alarm types and corresponding instances are arranged one under the other in the alarm editor.

Locking the data for an alarm type

You enter texts and attributes for event-dependent alarms in the alarm editor. While you are doing this, you can, for example, specify how the alarms are displayed on certain display devices (for example using the alarm class). Use the alarm types as templates to facilitate creation of alarms.

- When you enter data (attributes and texts) for the alarm type, you can specify whether or not they are locked. If you lock attributes, the icon of a closed chain link is shown beside the input box. If the attribute is not locked, the chain link is open.
- If you lock data in the alarm type, you can no longer change this in the alarms of the specific instance. The data is simply displayed.
- If, however, you want to make changes, you will need to go back to the alarm type, cancel the lock and make any changes there. The changes do not, however, apply to instances generated prior to the change.

Changing the data for an alarm type

If you change data for an alarm type, these changes are automatically included in the instances.
Exceptions: You have already changed this data in the instance or have locked or unlocked data later in the alarm type.

Note

If you copy instances to a different program without copying the alarm type, then the instance will not be completely displayed. In this case, copy the alarm type to the new program as well.

Resetting instance-specific data to the value of the alarm type

If attributes or texts were overwritten in an alarm instance, a type icon is displayed beside the attribute. You can decide whether to use the value of the alarm type again for each attribute. In this case, no type icon is displayed.

See also

Creating and editing an alarm type (Page 1189)

Creating and editing an instance DB (Page 1189)

9.5.6 Formal parameters, alarm data types and alarm blocks

Formal parameters as alarm number inputs

For each alarm or group of alarms, you require a formal parameter (name of the alarm) in your program that you specify as an IN parameter in the tag overview of your program. The formal parameter is used as an alarm number input and forms the basis for an alarm.

Supplying the formal parameter with the suitable data type

The formal parameter must be assigned an alarm data type in keeping with the alarm block being used.

Alarm data types and corresponding alarm blocks

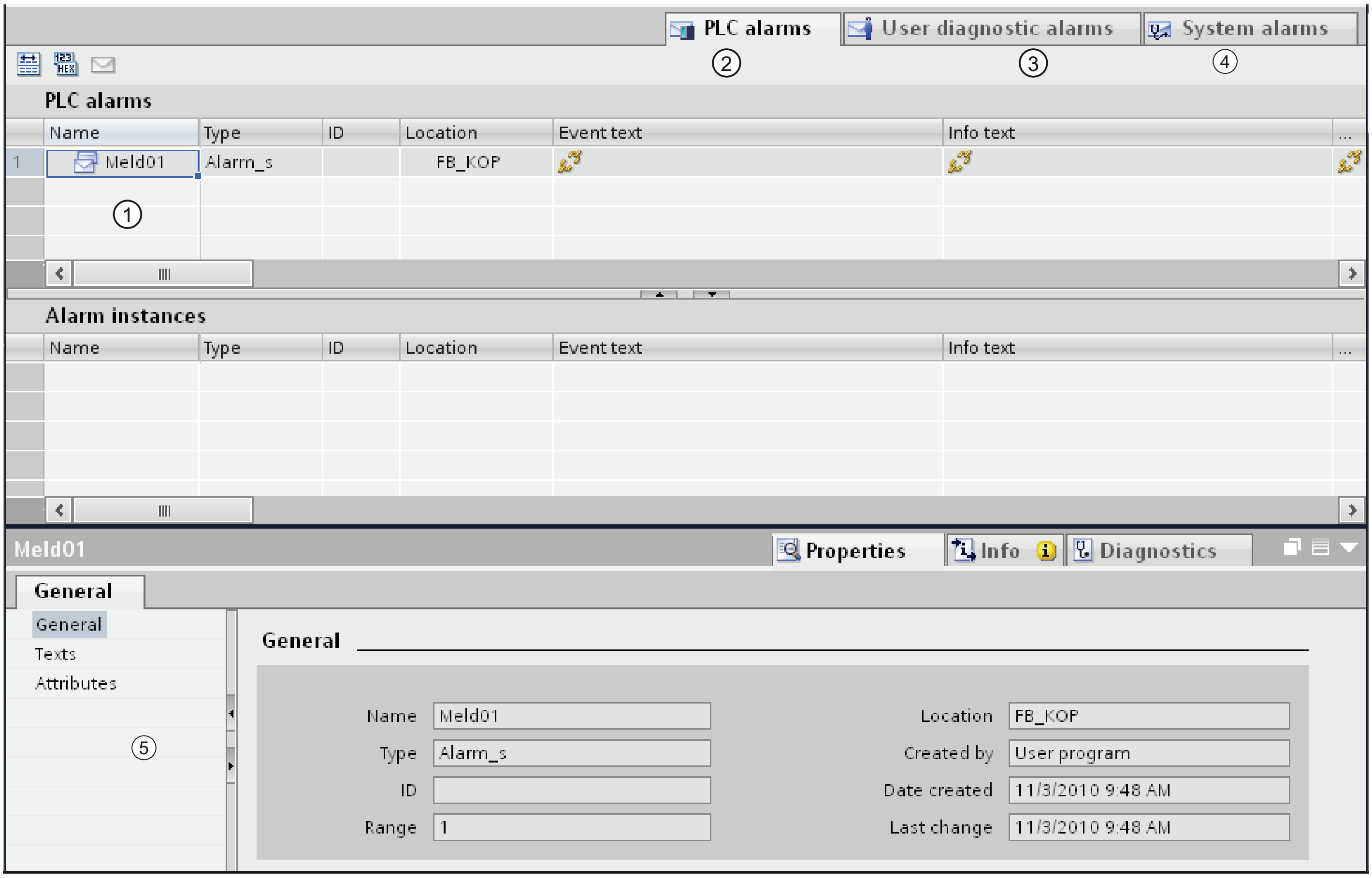
The following table shows the alarm data types with their corresponding alarm blocks and their properties. The values of the data types have the same names as those of the alarm blocks (exception: "alarm_s") and and have the prefix "C_".

Data type	Alarm block	Properties
C_Alarm_8	ALARM_8	8 channels, acknowledgment possible, no associated values
C_Alarm_8p	ALARM_8P	8 channels, acknowledgment possible, up to 10 associated values per channel
C_Notify	NOTIFY	1 channel, no acknowledgment, up to 10 associated values
C_Alarm	ALARM	1 channel, acknowledgment possible, up to 10 associated values
C_Alarm_s	ALARM_S	1 channel, no acknowledgment, up to 1 associated value
C_Alarm_s	ALARM_SQ	1 channel, acknowledgment possible, up to 1 associated value
C_Ar_Send	AR_SEND	Used to send an archive
C_Notify_8p	NOTIFY_8P	8 channels, no acknowledgment, up to 10 associated values
C_Alarm_s	ALARM_DQ	1 channel, acknowledgment possible, up to 1 associated value
C_Alarm_s	ALARM_D	1 channel, no acknowledgment possible, up to 1 associated value

9.5.7 Layout of the alarm editor

Layout of the alarm editor

The following graphic shows the components of the alarm editor:



PLC alarms

	Name	Type	ID	Location	Event text	Info text	...
1	Meld01	Alarm_s		FB_KOP			

Alarm instances

Name	Type	ID	Location	Event text	Info text	...

General

Name	Meld01	Location	FB_KOP
Type	Alarm_s	Created by	User program
ID		Date created	11/3/2010 9:48 AM
Range	1	Last change	11/3/2010 9:48 AM

①	Table showing the alarms in the work area.
②	"PLC alarms" tab: You can edit PLC alarms here.
③	"User diagnostic alarms" tab: You can create and edit user diagnostic alarms here.
④	"System alarms" tab: System alarms can only be viewed and cannot be edited.
⑤	Inspector window

You can enter or modify the texts and attributes in the table or in the Inspector window.

9.5.8 Creating and editing alarms

9.5.8.1 Creating PLC alarms

Prerequisites

You have created a function block.

Procedure

To configure a PLC alarm, follow these steps:

1. Select the function block (FB) for which you want to create a PLC alarm in the "Program blocks" folder in the project navigation and double-click on the block to open it.
2. Fill in the block interface. For each alarm block that will be called in the FB, you will need to declare tags in the calling FB.
   To achieve this for example, enter the following variables:
   – For the "IN" parameter, a name for the alarm block input, for example "Alarm01" (for alarm input 01) and the data type.
3. In the instruction window of the FB, insert the call for the selected alarm block, e.g. "CALL ALARM_S" and complete your input with the RETURN key.

   Result: The statement part of the FB displays the input tags of the called alarm block, in this case the ALARM_S block.
4. Assign the name for the alarm block input you named in step 2 to the "EV_ID" tag, in this case "Alarm01".

   Note

   If, instead of an alarm block in the CPU, you call an FB with multiple instances in which alarms are also configured, you would also need to configure the alarms of the FB with multiple instances in the calling block.
5. Repeat steps 2 through 4 for all alarm block calls in this FB.

9.5.8.2 Edit PLC alarms in the alarm editor

Prerequisites

You have created a PLC alarm.

Procedure

To edit PLC alarms, follow these steps:

1. Double-click "PLC alarms" in the project navigation. The alarm editor opens.
2. Enter the required texts and attributes in the appropriate columns.

> **Note**
>
> When you edit alarm types, you can lock texts and attributes. You can do this by clicking on the icon in front of the relevant column.
>
> If you edit alarms in instance DBs in which the texts/attributes are not locked in the alarm type, a type icon is displayed in front of the relevant column. When you click on this icon, modified texts/attributes are reset to the values of the alarm type.

9.5.8.3 Edit PLC alarms in the program editor

Prerequisites

You have created a PLC alarm.

Procedure

To edit PLC alarms, follow these steps:

1. Select the appropriate line in the block interface.
2. Move to the "Alarm" tab in the inspector window and select the required group.
3. Enter the required texts and attributes in the appropriate fields.

> **Note**
>
> When you edit alarm types, you can lock texts and attributes. You can do this by clicking on the icon next to the relevant field.
>
> If you edit alarms in instance DBs in which the texts/attributes are not locked in the alarm type, a type icon is displayed next to the relevant field. When you click on this icon, modified texts/attributes are reset to the values of the alarm type.

9.5.8.4 Deleting PLC alarms

Procedure

To delete a PLC alarm, follow these steps:

1. Open the block containing the alarm you want to delete.
2. Select the corresponding line in the block interface and select "Delete" in the shortcut menu.

Result

The alarm is deleted.

9.5.8.5 Creating and editing an alarm type

Procedure

To edit an alarm type, follow these steps:

1. Select the required alarm block.
2. Enter the texts you require in the appropriate columns or select the required attributes. If you have selected a multichannel alarm block (for example, "ALARM_8"), you can assign a separate text and certain separate attributes to each subnumber.
3. If you do not want the texts or attributes of the instance to be changed, lock them in the alarm type.

9.5.8.6 Creating and editing an instance DB

Prerequisites

You have already created an FB and created at least one alarm in it.

Procedure

To assign instance data blocks (DBs) to an alarm type and to edit the alarms for these DBs for specific instances, follow the steps below:

1. Double-click on "Add new block" in the project navigation, click on the "Data block (BD)" button that appears in the dialog and select the function block (alarm type) to which you want to assign the instance DB from the "Type" drop-down list.

2. Now click in the inspector window on the "Alarm" tab and select the group you want.
   Or:
   Double-click in the project navigation on "PLC alarms" to open the alarm configuration.
3. Make the changes you require for the particular instance DB.

> **Note**
>
> If the properties of the instance DB are write-protected then you must first unlock them in alarm type (FB).

Result

This completes the alarm configuration for the selected instance DB.

9.5.8.7 Creating user diagnostic messages

User diagnostic alarms are assigned to a CPU. They are created and edited in the alarm editor.

Procedure

To configure a user diagnostic alarm, follow these steps:

1. Double-click on the "PLC alarms" folder in project navigation to open the alarm editor.
2. Select the "User diagnostic alarms" tab in the alarm editor.
3. Click in the table and select "Insert new alarm" in the shortcut menu.

Result

You have created a user diagnostic alarm.

9.5.8.8 Editing user diagnostic alarms

Prerequisites

You have created a user diagnostic alarm.

The alarm editor is open.

Procedure

To edit a user diagnostic alarm, follow these steps:

1. Enter the required texts and attributes in the appropriate columns.

9.5.8.9 Deleting user diagnostic messages

You can delete a selected alarm. The texts you configured for this alarm are deleted and the alarm number is released for use.

Procedure

Follow the steps below to delete a user diagnostic message:

1. Select the corresponding row in the table and select "Delete" in the shortcut menu.

Result

The alarm is deleted. It is no longer displayed in the table.

9.5.8.10 Entering texts

You can enter the texts for alarms manually or you can use the default values.

Text template from the alarm type

All the texts in the alarm type are available as a template for creating alarm texts. If the alarm type already contains a general text, all instances of this alarm type include the same attributes and texts. Where necessary, you simply need to modify them.

Info text

The info text is a text that can be specified for certain display devices. With certain groups of devices (for example WinCC), it can be changed during run time.

Additional texts

Additional texts are texts that can be displayed by certain HMI devices. Click in the relevant row and type in the text. If you want to protect the text from being overwritten, click the option in the column. The texts can include line breaks.

See also

Locking texts (Page 1192)

9.5.8.11 Locking texts

"Locked" option in the alarm type

You can only lock texts when you edit the alarm type. The locked texts in alarms derived from the alarm type are write-protected. The icon displayed beside the input field indicates whether or not they are locked.

Locking texts

Follow the steps below to lock texts:

1. Start by editing the alarm types.
2. Click on the icon beside the input box you want to lock.

   Result: The icon changes to a closed chain link.

Unlocking texts

To unlock texts, follow the steps below:

1. Start by editing the alarm types.
2. Click on the icon beside the input box you want to unlock.

   Result: The icon changes to an open chain link.

9.5.8.12 Locking attributes

Locking attributes in the alarm type

You can only lock attributes when you edit the alarm type. The locked attributes in alarms derived from the alarm type are write-protected. The icon displayed in front of the input field indicates whether or not they are locked.

Locking attributes

Follow the steps below to lock attributes:

1. Start by editing the alarm types.
2. Click on the icon to the left of the input box you want to lock in the table.

   Result: The icon changes to a closed chain link.

Unlocking attributes

Follow the steps below to unlock attributes:

1. Start by editing the alarm types.
2. Click on the icon to the left of the input box you want to unlock in the table.

   Result: The icon changes to an open chain link.

9.5.8.13 Insert associated values in alarms

To add current information to alarms, e.g. from the process, you can insert associated values at any location within an alarm text.

Procedure

To insert an associated value into an alarm, follow these steps:

1. Configure a block as follows:
   @<No. of associated value><Element type><Format>@.
2. Insert this block at the locations in the alarm text at which the associated value is to be shown.

See also

Structure of associated values (Page 1194)

Examples of associated values (Page 1195)

9.5.8.14 Structure of associated values

Associated values are comprised of the following:

Element type

This uniquely configures the data type of the associated value:

Element type	Data type
Y	BYTE
W	WORD
X	DWORD
I	Integer
D	DINT
B	BOOL
C	CHAR
R	REAL

The element type only uniquely identifies the data type transferred by the AS. It is not used as Casting Operator.

Format

Determine the output format for the associated value on the display device. The format is preceded by the "%" sign. The following fixed formats apply to alarm texts:

Format	Description
%[i]X	Hexadecimal number with i digits
%[i]u	Decimal number without sign with i digits
%[i]d	Decimal number with sign with i digits
%[i]b	Binary number with i digits
%[i][.y]f	Fixed number of points Signed value of the form dddd: one or more numbers with y digits after the decimal point and total number of digits i
%[i]s	String (ANSI string) with i digits Characters are printed up to the first 0 Byte (00Hex).
%t#<Name of text library>	Access to text library

If the format is too small then the value is nevertheless output in full.

If the format is too large then an appropriate number of empty characters are output before the value.

Note

Please note that you can optionally enter "[i]", without the square brackets.

See also

Insert associated values in alarms (Page 1193)

Examples of associated values (Page 1195)

9.5.8.15 Examples of associated values

Examples of associated values:

@1I%6d@: The value from associated value 1 is shown as a decimal number with a max. 6 digits.

@2R%6f@: The value "5.4", for instance from associated value 2, is shown as a fixed number of points "5.4" (three leading blanks).

@2R%2f@: The value "5.4", for instance from associated value 2, is shown as a fixed number of points "5.4" (if number of digits is too small then these are not cut off).

@1W%t#Textbib1@: Associated value 1 of the WORD data type is the index by which the text to be inserted is referenced in the Textbib1 text library.

Note

If you want to transfer more than one associated value to an ALARM_S block then you can transfer an array with a max. length of 12 bytes. These could be, e.g. maximum 12 Byte or Char, maximum 6 Word or Int or maximum 3 DWord, Real or Dint.

See also

Insert associated values in alarms (Page 1193)

Structure of associated values (Page 1194)

9.5.8.16 Deleting associated values

You can delete associated values by deleting the string that represents an associated value in the alarm text.

Procedure:

Follow the steps below to delete associated values:

1. Find the block in the alarm text that corresponds to the associated value you want to delete.
   The block begins with the "@" character followed by the location ID by which you can recognize the associated value, there is then format information and it ends with the "@" character.
2. Delete the block you have found from the alarm text.

9.5.9 Text lists for alarms

9.5.9.1 Basics of alarm text lists

You can adapt existing text lists (user-defined and system-defined text lists) to your requirements and edit texts and attributes. You can then translate the texts into the project language(s) you want to use.

Detailed information on text lists is contained in the "Working with text lists" chapter.

9.5.9.2 Editing text lists for alarms

Requirement

- The user interface language and the project language must be the same during editing.

Procedure

Follow the steps below to edit text lists:

1. Double-click on the "Text lists" command below "Common data" in project navigation or select the context command "Go to text list" in the alarm editor.

   The text list editor opens.
2. Select the text list you want to edit from the table.
3. Change the values as required.

You can change the following values:

- Text titles:
  With the exception of the Info text, the titles for the alarm texts (alarm text, additional texts) can be freely configured to suit your purposes.
- Names of attribute values:
  Many of the names for attribute values (for example Priority, Display class, ...) can be configured freely. The index in the system text list matches the index in the selection box of the corresponding attribute in the alarm editor.
  Example: If you enter the text "Priority_0" at index 0 in the priority text list, the same text will be displayed at the first position in the selection box in the alarm editor.

9.5.9.3 Integrating texts from text lists in alarms

You can integrate any number of texts in an alarm from various text lists. The texts can be positioned anywhere you want which means that they can be used in alarms in foreign languages.

Procedure

To integrate texts from text lists in alarms, follow the steps below:

1. Double-click on the "Text lists" command below the PLC in the program navigation.

   The text list editor opens
2. Find the index of the text you want to integrate.
3. Put a placeholder in the format @[Index]%t#[textlist]@ at the point at which you want the text to appear.

Note

[Index] = for example, 1W, where 1W is the first associated value of the alarm of the type WORD.

9.5.9.4 Example of integrating texts from text lists in alarms

Configured alarm text: Pressure has risen @2W%t#textlist1@.

Text list with the name "textlist1":

Index	German	English
1734	zu hoch	too high

The associated value is supplied with the value 1734.

The following alarm text is displayed: Pressure has risen too high.

See also

Integrating texts from text lists in alarms (Page 1197)

9.5.10 Alarm classes

9.5.10.1 Creating alarm classes

You can configure alarm classes to suit your purposes. You can create and edit them in the alarm class editor. An alarm can then by assigned to an alarm class in the alarm editor.

Prerequisites

You have opened the "Common data" folder in project navigation.

Procedure

To configure an alarm class, follow these steps:

1. Double click on the "Alarm classes" entry in project navigation.

   The alarm class editor opens.
2. Select "Insert new alarm class" in the shortcut menu.
3. Enter a unique name for the new alarm class in the "Name" column.

   The language of the name you assign here is neutral.
4. Enter a display name in the "Display name" column. This name is translatable.
5. Select whether or not this alarm class requires acknowledgment in the "With acknowledgment" column.

9.5.10.2 Editing alarm classes

You can change the settings (name, display name, or acknowledgment) for an alarm class at any time even if alarms have already been assigned to the alarm class. The changes are adopted automatically in the alarms.

Copying alarm classes

To copy alarm classes, follow these steps:

1. Select the row with the alarm class you want to copy.
2. Select "Copy" in the shortcut menu.
3. Select "Paste" in the shortcut menu.

Result

The copied alarm class is appended to the end of the table under a new name.

The name of the new alarm class is made up as follows:

<old name><no.>

no.: This the lowest free natural number.

9.6 Using global project functions

9.6.1 Importing and exporting

9.6.1.1 Basics for importing and exporting

Introduction

You can export PLC tag tables to a standardized XLSX format for editing with external table editors. Similarly, you can import PLC tag tables created with external table editors to the TIA Portal.

Overwriting existing PLC tags and constants during import

Existing entries of the same name will be overwritten during import if they have the same name as the entries that will be imported.

Link to existing objects

References to PLC tags or constants that already exist in the project are updated automatically during import. The update is executed based on the name of the PLC tags and constants.

See also

Format of the export file (Page 1200)

Exporting PLC tags (Page 1201)

Importing PLC tags (Page 1202)

9.6.1.2 Format of the export file

Introduction

During the export of PLC tag tables, a standardized XSLX format will be generated that you can edit with external table editors.

This format is also expected during the import of tables.

Format of the export file

The sheet is always named "PLC Tags". This sheet can contain the displayed columns. The sorting order of columns can vary. The sheet does not necessarily have to include all columns. During import, the following values will be identified by a <no value> entry.

The names of the column headers are also clearly defined and are always expected in English.

The following table specifies the content expected for the individual columns:

Element	Explanation
Name	Name of the tags
Path	Group and name of the PLC tag table
Data type	The notation of the data type corresponds to the notation used in the PLC tag table.
Address	The address can be specified with German or international mnemonics.
Comment	Free-form comments
Visible in HMI	The value TRUE or FALSE is expected.
Accessible from HMI	The value TRUE or FALSE is expected.

See also

Basics for importing and exporting (Page 1200)

Exporting PLC tags (Page 1201)

Importing PLC tags (Page 1202)

9.6.1.3 Exporting PLC tags

Requirement

A PLC tag table is open.

Procedure

To export PLC tags and constants, follow these steps:

1. In the PLC tag table, click the "Export" button.

   The "Export to Excel" dialog opens.
2. Select the path to which you want to save the export file.
3. Select whether to export tags and/or constants.
4. Click the "OK" button.

Result

The export file will be generated. Errors and warnings generated during export are indicated in the "Info" tab of the Inspector window.

See also

Basics for importing and exporting (Page 1200)

Format of the export file (Page 1200)

Importing PLC tags (Page 1202)

9.6.1.4 Importing PLC tags

Requirement

A table exists and it conforms to format specifications.

Procedure

To import a PLC tag table, follow these steps:

1. Open the "All tags" table.
2. Click the "Import" button.

   The "Import from Excel" dialog opens.
3. Select whether to import PLC tags and/or constants.
4. Select the table you want to import.
5. Click the "OK" button.

Result

The PLC tag table will be imported.

Errors and warnings generated during export are indicated in the "Info" tab of the Inspector window.

See also

Basics for importing and exporting (Page 1200)

Format of the export file (Page 1200)

Exporting PLC tags (Page 1201)

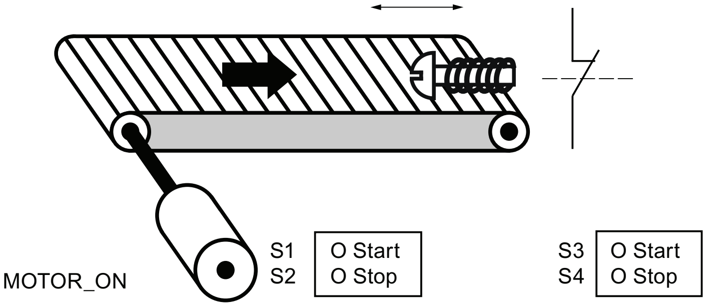
9.7 Programming examples

9.7.1 LAD programming examples

9.7.1.1 Example of controlling a conveyor belt

Controlling a conveyor belt

The following figure shows a conveyor belt that can be activated electrically. There are two push-button switches at the beginning of the belt: S1 for Start and S2 for Stop. There are also two push-button switches at the end of the belt: S3 for Start and S4 for Stop. It is possible to start or stop the belt from either end.



Implementation

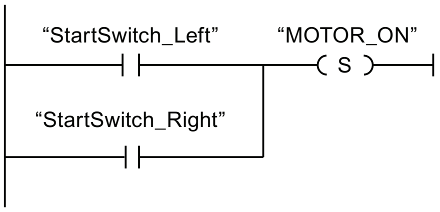
The following table shows the definition of the tags used:

Name	Data type	Comment
Startschalter_Links (S1)	Bool	Start switch on the left side of the conveyor belt
Stoppschalter_Links (S2)	Bool	Stop switch on the left side of the conveyor belt
Startschalter_Rechts (S3)	Bool	Start switch on the right side of the conveyor belt
Stoppschalter_Rechts (S4)	Bool	Stop switch on the right side of the conveyor belt
MOTOR_ON	Bool	Turn on the conveyor belt motor

The following networks show the LAD programming for solving this task:

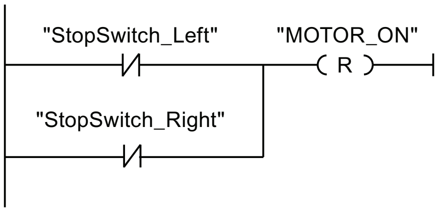
Network 1:

The conveyor belt motor is switched on when Start switch "S1" or "S3" is pressed.



Network 2:

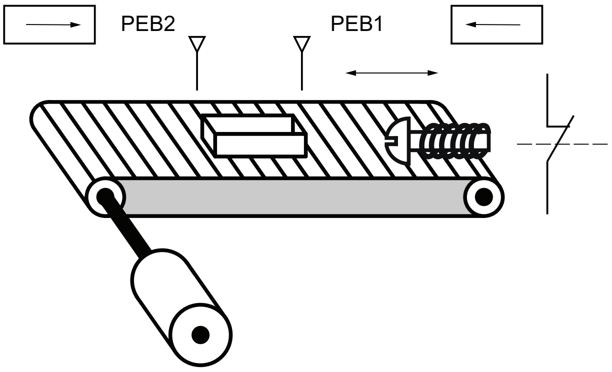
The conveyor belt motor is switched off when Stop switch "S2" or "S4" is pressed.



9.7.1.2 Example of detecting the direction of a conveyor belt

Detecting the direction of a conveyor belt

The following figure shows a conveyor belt that is equipped with two photoelectric barriers (PEB1 and PEB2). These are designed to detect the direction in which a package is moving on the belt.



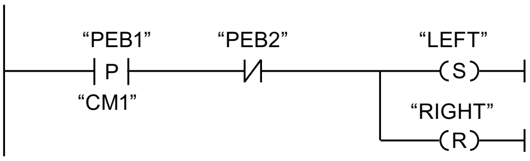
Implementation

The following table shows the definition of the tags used:

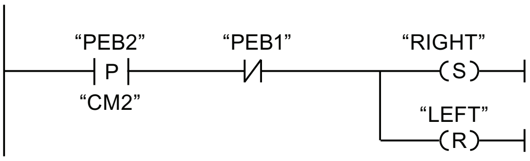
Name	Data type	Comment
PEB1	Bool	Photoelectric barrier 1
PEB2	Bool	Photoelectric barrier 2
RIGHT	Bool	Display during movement to right
LEFT	Bool	Display during movement to left
CM1	Bool	Edge bit memory 1
CM2	Bool	Edge bit memory 2

The following networks show the LAD programming for solving this task:

Network 1: If the signal changes from "0" to "1" (positive edge) at "PEB1" and, at the same time, the signal state at "PEB2" is "0", then the package on the belt is moving to the left.



Network 2: If the signal changes from "0" to "1" (positive edge) at "PEB2" and, at the same time, the signal state at "PEB1" is "0", then the package on the belt is moving to the right.

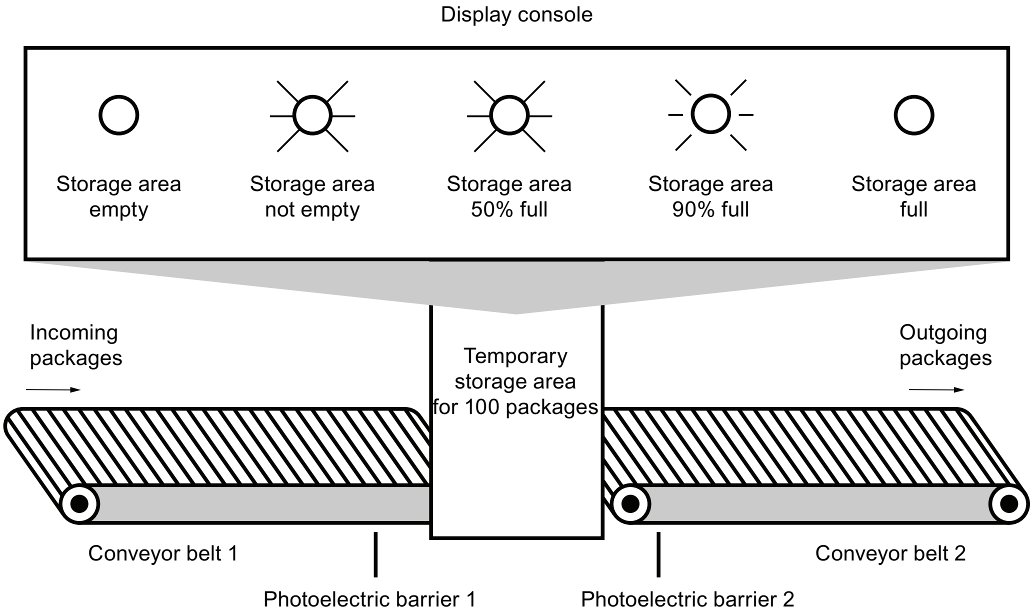


9.7.1.3 Example of detecting the fill level of a storage area

Detecting the fill level of a storage area

The following figure shows a system with two conveyor belts and a temporary storage area between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier at the end of conveyor belt 1 near the storage area detects how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock onto which the packages are loaded for delivery to customers by truck. A photoelectric barrier at the storage area exit detects how many packages leave the storage area to go to the loading dock. A display panel with five lamps indicates the utilization of the temporary storage area.

When a conveyor belt is restarted, the current count value is set to the number of packages available in the storage area.



Implementation

The following table shows the definition of the tags used:

Name	Data type	Comment
PEB1	BOOL	Photoelectric barrier 1
PEB2	BOOL	Photoelectric barrier 2
RESET	BOOL	Reset counter
LOAD	BOOL	Set counter to value of "PV" parameter
STOCK	INT	Stock at restart
PACKAGECOUNT	INT	Number of packages in the storage area (current count value)
STOCK_PACKAGE	BOOL	Is set if the current count value is greater than or equal to the value of the tag "STOCK".
STOR_EMPTY	BOOL	Display lamp: Storage area empty
STOR_NOT_EMPTY	BOOL	Display lamp: Storage area not empty
STOR_50%_FULL	BOOL	Display lamp: Storage area 50 % full
STOR_90%_FULL	BOOL	Display lamp: Storage area 90% full
STOR_FULL	BOOL	Display lamp: Storage area full
VOLUME_50	INT	Comparison value: 50 packages
VOLUME_90	INT	Comparison value: 90 packages
VOLUME_100	INT	Comparison value: 100 packages

The following networks show the LAD programming for activating the lamps:

Network 1:

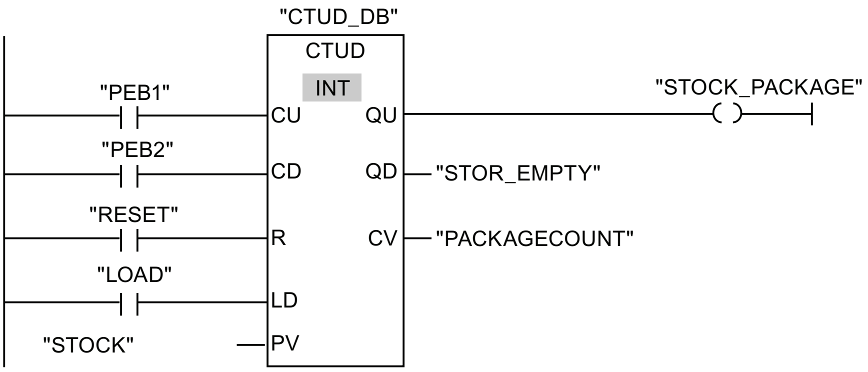
When a package is delivered to the storage area, the signal state at "PEB1" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB1", the "Up" counter is enabled, and the current count value of "PACKAGECOUNT" is increased by one.

When a package is delivered from the storage area to the loading dock, the signal state at "PEB2" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB2", the "Down" counter is enabled, and the current count value of "PACKAGECOUNT" is decreased by one.

If there are no packages in the storage area ("PACKAGECOUNT" = "0"), the "STOR_EMPTY" tag is set to signal state "1", and the "Storage area empty" lamp is switched on.

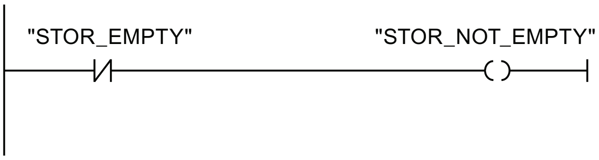
The current count value can be reset to "0" if the "RESET" tag is set to signal state "1".

If the "LOAD" tag is set to signal state "1", the current count value is set to the value of the "STOCK" tag. If the current count value is greater than or equal to the value of the "STOCK" tag , the "STOCK_PACKAGE" tag supplies the signal state "1".



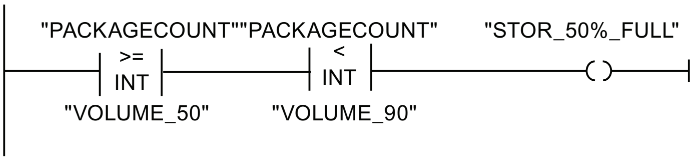
Network 2:

As long as there are packages in the storage area the "STOR_NOT_EMPTY" tag is set to signal state "1", and the "Storage area not empty" lamp is switched on.



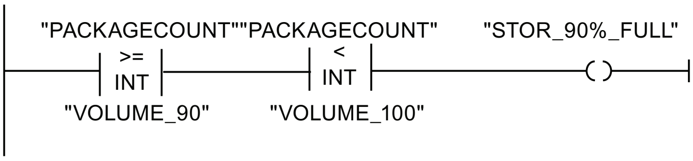
Network 3:

If the number of packages in the storage area is greater than or equal to 50, the "Storage area 50 % full" lamp switches on.



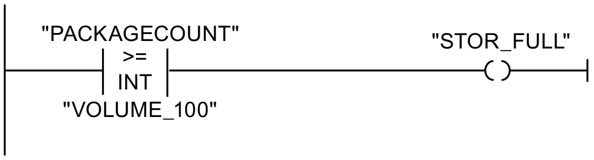
Network 4:

If the number of packages in the storage area is greater than or equal to 90, the "Storage area 90% full" lamp switches on.



Network 5:

If the number of packages in the storage area reaches 100, the "Storage area full" lamp switches on.



9.7.1.4 Example of controlling room temperature

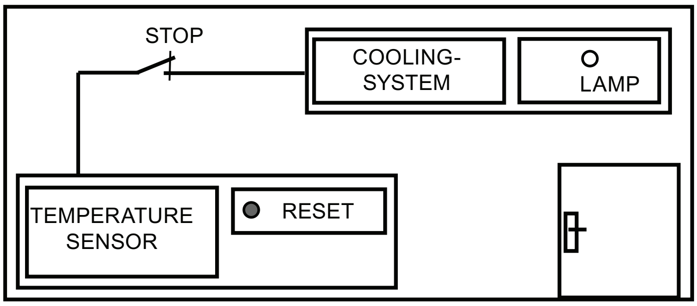
Controlling room temperature

In a cold room, the temperature must be maintained below zero degrees Celsius. Temperature fluctuations are monitored by means of a sensor. If the temperature rises above zero degrees Celsius, the cooling system switches on for a predetermined time. The "Cooling system On" lamp is lit during this time.

The cooling system and the lamp are turned off if one of the following conditions is fulfilled:

- The sensor reports a temperature fall below zero degrees Celsius.
- The preset cooling time has elapsed.
- The pushbutton switch "Stop" has been pressed.

If the preset cooling time has expired, and the temperature in the cold room is still too high, the cooling system can be restarted by means of the pushbutton switch "Reset".



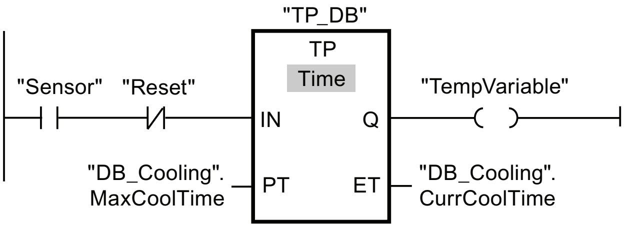
Implementation

The following table shows the definition of the tags used:

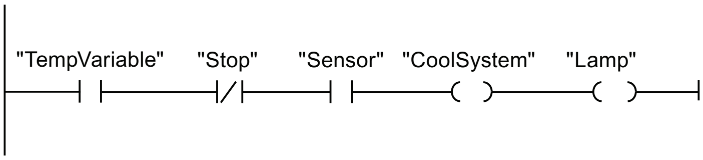
Name	Data type	Comment
Sensor	BOOL	Temperature sensor signal
Reset	BOOL	Restart
Stop	BOOL	The cooling system is switched off.
MaxCoolTime	TIME	Predetermined cooling time This tag is defined in the "DB_Cool" data block.
CurrCoolTime	TIME	Currently elapsed cooling time This tag is defined in the "DB_Cool" data block.
CoolSystem	BOOL	The cooling system is switched on.
Lamp	BOOL	The "Cooling system On" lamp is switched on.
TempVariable	BOOL	Temporary variable This tag stores the signal status of the IEC timer TP.

The following network shows the LAD programming for controlling room temperature:

Network 1:



Network 2:



When the temperature in the cold room rises above zero degrees Celsius, the signal state at the "Sensor" operand switches from "0" to "1" (positive signal edge). In case of a positive signal edge at the input IN, the timer function for the preset cooling time is started and the "TempVariable" receives the signal state "1". The signal state "1" of the "TempVariable" causes the cooling system and the display lamp to be turned on in network 2. The outputs "Sensor", "Cooling system" and "Lamp" must be programmed in network 2, because you can program only one coil at output Q of the timer function.

If the temperature in the cold room falls below zero degrees Celsius, the signal state of the sensor switches back to "0". This switches the cooling system and lamp off.

If the sensor does not signal a temperature drop, the cooling system and lamp are switched off after the predetermined cooling time has elapsed, at the latest. In this case, the cooling process can be restarted by pressing the "Reset" push-button switch. Pressing and releasing the push-button switch generates a new positive signal edge at input IN, which restarts the cooling system.

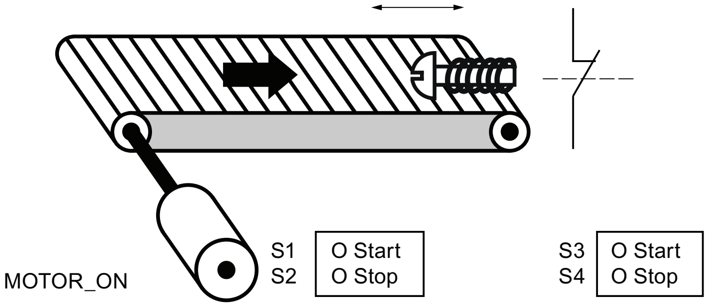
The cooling system and the display lamps can be turned off with the pushbutton switch "Stop" at any time.

9.7.2 FBD programming examples

9.7.2.1 Example of controlling a conveyor belt

Controlling a conveyor belt

The following figure shows a conveyor belt that can be activated electrically. There are two push-button switches at the beginning of the belt: S1 for Start and S2 for Stop. There are also two push-button switches at the end of the belt: S3 for Start and S4 for Stop. It is possible to start or stop the belt from either end.



Implementation

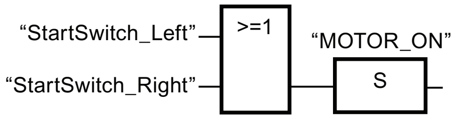
The following table shows the definition of the tags used:

Name	Data type	Comment
Startschalter_Links (S1)	Bool	Start switch on the left side of the conveyor belt
Stoppschalter_Links (S2)	Bool	Stop switch on the left side of the conveyor belt
Startschalter_Rechts (S3)	Bool	Start switch on the right side of the conveyor belt
Stoppschalter_Rechts (S4)	Bool	Stop switch on the right side of the conveyor belt
MOTOR_ON	Bool	Turn on the conveyor belt motor

The following networks show the FBD programming for accomplishing this task:

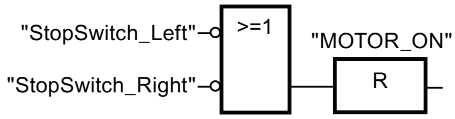
Network 1:

The conveyor belt motor is switched on when Start switch "S1" or "S3" is pressed.



Network 2:

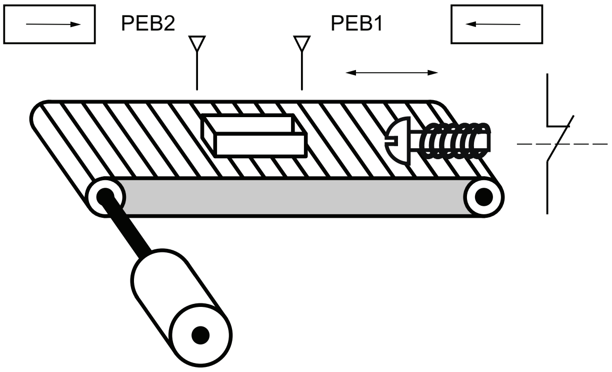
The conveyor belt motor is switched off when Stop switch "S2" or "S4" is pressed.



9.7.2.2 Example of detecting the direction of a conveyor belt

Detecting the direction of a conveyor belt

The following figure shows a conveyor belt that is equipped with two photoelectric barriers (PEB1 and PEB2). These are designed to detect the direction in which a package is moving on the belt.



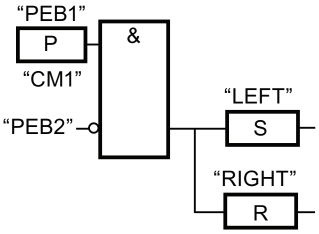
Implementation

The following table shows the definition of the tags used:

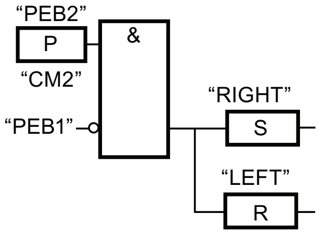
Name	Data type	Comment
PEB1	Bool	Photoelectric barrier 1
PEB2	Bool	Photoelectric barrier 2
RIGHT	Bool	Display during movement to right
LEFT	Bool	Display during movement to left
CM1	Bool	Edge bit memory 1
CM2	Bool	Edge bit memory 2

The following networks show the FBD programming for solving this task:

Network 1: If the signal changes from "0" to "1" (positive edge) at "PEB1" and, at the same time, the signal state at "PEB2" is "0", then the package on the belt is moving to the left.



Network 2: If the signal state changes from "0" to "1" (positive edge) at "PEB2" and, at the same time, the signal state at "PEB1" is "0", then the package on the belt is moving to the right.



9.7.2.3 Example of detecting the fill level of a storage area

Detecting the fill level of a storage area

The following figure shows a system with two conveyor belts and a temporary storage area between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier at the end of conveyor belt 1 near the storage area detects how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock onto which the packages are loaded for delivery to customers by truck. A photoelectric barrier at the storage area exit detects how many packages leave the storage area to go to the loading dock. A display panel with five lamps indicates the utilization of the temporary storage area.

When a conveyor belt is restarted, the current count value is set to the number of packages available in the storage area.

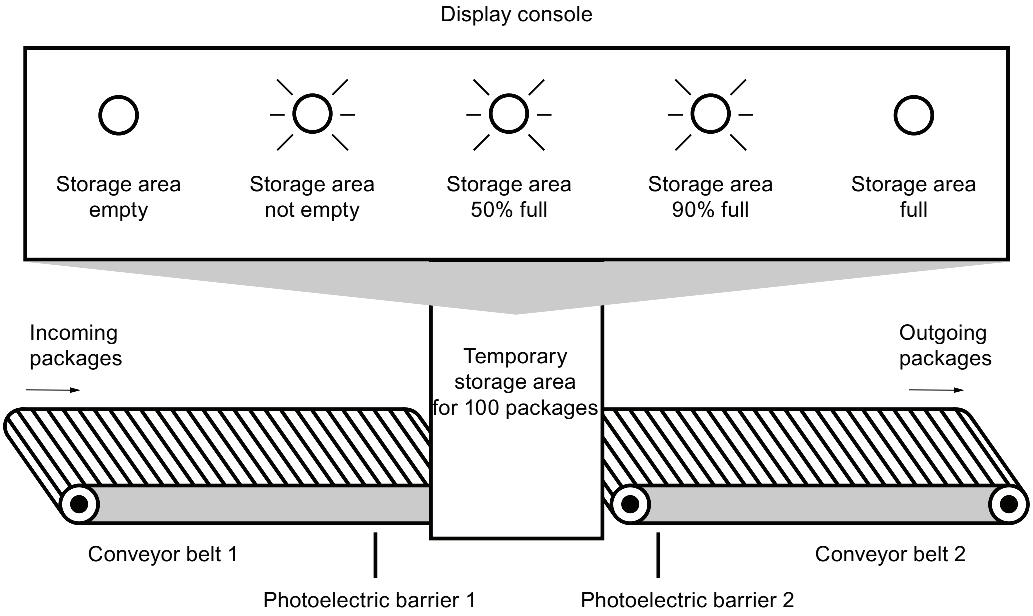


Figure 9-1

Implementation

The following table shows the definition of the tags used:

Name	Data type	Comment
PEB1	BOOL	Photoelectric barrier 1
PEB2	BOOL	Photoelectric barrier 2
RESET	BOOL	Reset counter
LOAD	BOOL	Set counter to value of "CV" parameter
STOCK	INT	Stock at restart
PACKAGECOUNT	INT	Number of packages in the storage area (current count value)
STOCK_PACKAGE	BOOL	Is set if the current count value is greater than or equal to the value of the tag "STOCK".
STOR_EMPTY	BOOL	Display lamp: Storage area empty
STOR_NOT_EMPTY	BOOL	Display lamp: Storage area not empty
STOR_50%_FULL	BOOL	Display lamp: Storage area 50 % full
STOR_90%_FULL	BOOL	Display lamp: Storage area 90% full
STOR_FULL	BOOL	Display lamp: Storage area full
VOLUME_50	INT	Comparison value: 50 packages
VOLUME_90	INT	Comparison value: 90 packages
VOLUME_100	INT	Comparison value: 100 packages

The following networks show the FBD programming for activating the lamps:

Network 1:

When a package is delivered to the storage area, the signal state at "PEB1" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB1", the "Up" counter is enabled, and the current count value of "PACKAGECOUNT" is increased by one.

When a package is delivered from the storage area to the loading dock, the signal state at "PEB2" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB2", the "Down" counter is enabled, and the current count value of "PACKAGECOUNT" is decreased by one.

If there are no packages in the storage area ("PACKAGECOUNT" = "0"), the "STOR_EMPTY" tag is set to signal state "1", and the "Storage area empty" lamp is switched on.

The current count value can be reset to "0" if the "RESET" tag is set to signal state "1".

If the "LOAD" tag is set to signal state "1", the current count value is set to the value of the "STOCK" tag. If the current count value is greater than or equal to the value of the "STOCK" tag , the "STOCK_PACKAGE" tag supplies the signal state "1".

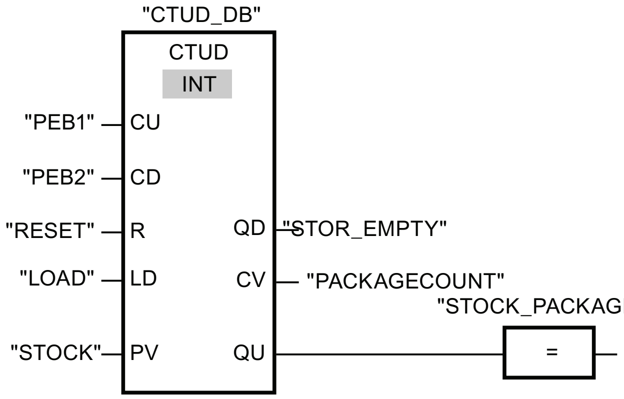
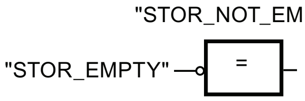


Figure 9-2

Network 2:

As long as there are packages in the storage area the "STOR_NOT_EMPTY" tag is set to signal state "1", and the "Storage area not empty" lamp is switched on.



Network 3:

If the number of packages in the storage area is greater than or equal to 50, the "Storage area 50 % full" lamp switches on.

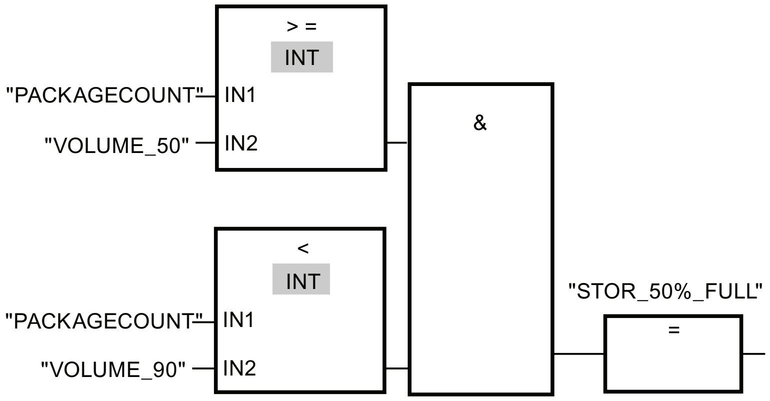


Figure 9-3

Network 4:

If the number of packages in the storage area is greater than or equal to 90, the "Storage area 90% full" lamp switches on.

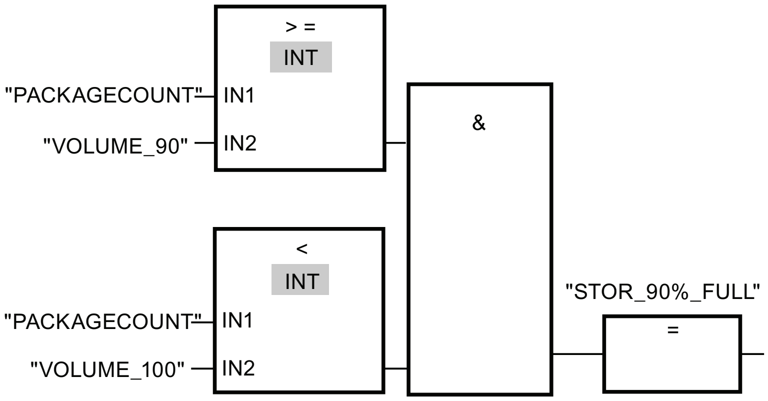
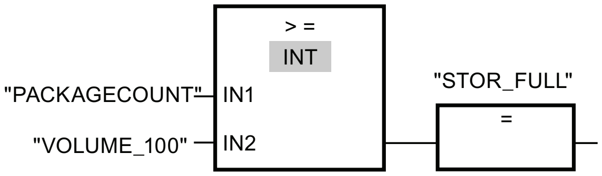


Figure 9-4

Network 5:

If the number of packages in the storage area reaches 100, the "Storage area full" lamp switches on.



9.7.2.4 Example of controlling room temperature

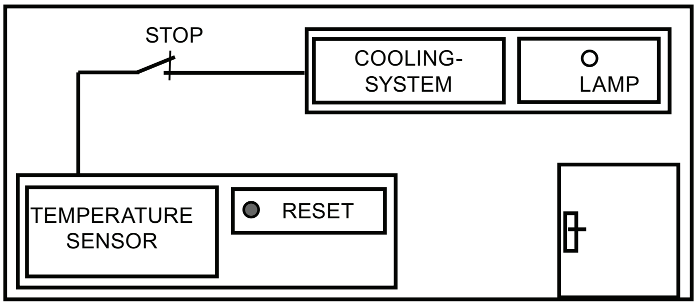
Controlling room temperature

In a cold room, the temperature must be maintained below zero degrees Celsius. Temperature fluctuations are monitored by means of a sensor. If the temperature rises above zero degrees Celsius, the cooling system switches on for a predetermined time. The "Cooling system On" lamp is lit during this time.

The cooling system and the lamp are turned off if one of the following conditions is fulfilled:

- The sensor reports a temperature fall below zero degrees Celsius.
- The preset cooling time has elapsed.
- The pushbutton switch "Stop" has been pressed.

If the preset cooling time has expired, and the temperature in the cold room is still too high, the cooling system can be restarted by means of the pushbutton switch "Reset".



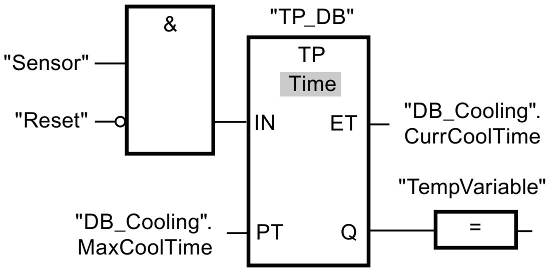
Implementation

The following table shows the definition of the tags used:

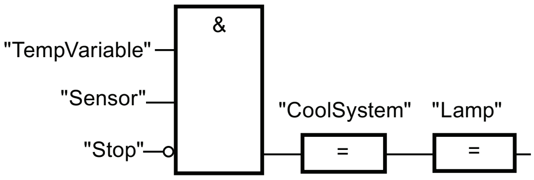
Name	Data type	Comment
Sensor	BOOL	Temperature sensor signal
Reset	BOOL	Restart
Stop	BOOL	The cooling system is switched off.
MaxCoolTime	TIME	Predetermined cooling time This tag is defined in the "DB_Cool" data block.
CurrCoolTime	TIME	Currently elapsed cooling time This tag is defined in the "DB_Cool" data block.
CoolSystem	BOOL	The cooling system is switched on.
Lamp	BOOL	The "Cooling system On" lamp is switched on.
TempVariable	BOOL	Temporary variable This tag stores the signal status of the IEC timer TP.

The following network shows the FBD programming for controlling room temperature:

Network 1:



Network 2:



When the temperature in the cold room rises above zero degrees Celsius, the signal state at the "Sensor" operand switches from "0" to "1" (positive signal edge). In case of a positive signal edge at the input IN, the timer function for the preset cooling time is started and the "TempVariable" receives the signal state "1". The signal state "1" of the "TempVariable" causes in network 2 that the cooling system as well as the display lamp are turned on. The outputs "Sensor", "Cooling system" and "Lamp" must be programmed in network 2, because you can program only one coil at output Q of the timer function.

If the temperature in the cold room falls below zero degrees Celsius, the signal state of the sensor switches back to "0". This switches the cooling system and lamp off.

If the sensor does not signal a temperature drop, the cooling system and lamp are switched off after the predetermined cooling time has elapsed, at the latest. In this case, the cooling process can be restarted by pressing the "Reset" push-button switch. Pressing and releasing the push-button switch generates a new positive signal edge at input IN, which restarts the cooling system.

The cooling system and the display lamps can be turned off with the pushbutton switch "Stop" at any time.

9.8 References

9.8.1 General parameters of the instructions

9.8.1.1 Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions

Asynchronous instructions

For instructions that work asynchronously, the function is executed with several calls.

Identification of the job

If you use asynchronous instructions to trigger a process interrupt, output control commands to DP slaves, start a data transfer, or abort a non-configured connection with one of the SFCs listed above and then call the same SFC again before the current job is completed, then the reaction of the SFC will depend on whether the second call involves the same job.

Parameter REQ

The input parameter REQ (request) is used solely to start the job:

- If you call the instruction for a job that is not currently active, the job is started with REQ = 1 (case 1).
- If a particular job has been started and not yet completed and you call the instruction again to perform the same job (for example, in a cyclic interrupt OB), then REQ is not evaluated by the instruction (case 2).

Parameter RET_VAL and BUSY

The output parameters RET_VAL and BUSY indicate the status of the job.

Pay attention to the note in section: Evaluating errors with output parameter RET_VAL (Page 1222)

- In case 1 (first call with REQ=1), the input parameter will be entered in RET_VALW#16#7001 if system resources are available and supply is correct. BUSY will be set.

  If the required system resources are currently being used or the input parameters have errors, the corresponding error code is entered in RET_VAL and BUSY has the value 0.
- In case 2 (interim call) W#16#7002 will be entered in RET_VAL (this corresponds to a warning: Job still being processed!), and BUSY will be set.
- The following applies to the last call for a job:
  - For instruction "DPNRM_DG (Page 1725)", the number of data in bytes will entered as integer in RET_VAL in case there are no errors in data transmission. BUSY has the value "0" in this case.

    If there is an error, then the error information will be entered in RET_VAL and you should not evaluate BUSY in this case.
  - For all other instructions, "0" will be entered in RET_VAL if the job was executed without errors and BUSY has the value "0" in this case. If there is an error, the error code is entered in RET_VAL and BUSY has the value "0" in this case.

Note

If the first and last call coincide, the reaction is the same for RET_VAL and BUSY as described for the last call.

Overview

The following table provides you with an overview of the relationships explained above. In particular, it shows the possible values of the output parameters if the execution of the job is not completed after an instruction call has been completed.

Note

Following every call, you must evaluate the relevant output parameters in your program.

Relationship between call, REQ, RET_VAL and BUSY during execution of a "running" job.

Number of the call	Type of call	REQ	RET_VAL	BUSY
1	First call	1	W#16#7001	1
			Error code	0
2 to (n - 1)	Intermediate call	irrelevant	W#16#7002	1

Number of the call	Type of call	REQ	RET_VAL	BUSY
n	Last call	irrelevant	W#16#0000, if no errors have occurred.	0
			Error code if errors occurred	0

9.8.1.2 Evaluating errors with output parameter RET_VAL

Types of error information

An executed instruction indicates in the user program whether or not the CPU was able to execute the function of the instruction successfully.

You can obtain information about any errors that occurred in two ways:

- In the BR bit of the status word
- in the output parameter RET_VAL (return value).

Note

Before evaluating the output parameters specific to an instruction, you should always follow the steps below:

- First, evaluate the BR bit of the status word.
- Then check the output parameter RET_VAL.

If the BR bit indicates that an error has occurred or if RET_VAL contains a general error code, you should not evaluate the instruction-specific output parameters.

Error information in the return value

An instruction indicates that an error occurred during its execution by entering the value "0" in the binary result bit (BR) of the status word. Some instructions provide an additional error code at an output parameter known as the return value (RET_VAL). If a general error is entered in the output parameter RET_VAL (see below for explanation), this is only indicated by the value "0" in the BR bit of the status word.

The return value is of the data type integer (INT). The relationship of the return value to the value "0" indicates whether or not an error occurred during execution of the function.

CPU execution of the instruction	BR	Return value	Sign of the integer
With error(s)	0	less than "0"	negative (sign bit is "1")
Without error	1	greater than or equal to "0"	positive (sign bit is "0")

Reacting to error information

There are two types of error codes in RET_VAL:

- A general error code that all instructions can output and
- A specific error code that an instruction can output and which relates to its specific function.

You can write your program so that it reacts to the errors that occur during execution of an instruction. This way you prevent further errors occurring as a result of the first error.

General and specific error information

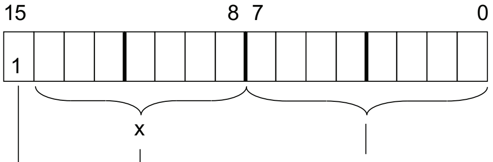
The return value (RET_VAL) of an instruction provides one of the two following types of error codes:

- A general error code that relates to errors that can occur in any instruction.
- A specific error code that relates only to the particular instruction.

Even though the data type of the output parameter RET_VAL is an integer (INT), the error codes for the instruction are grouped according to hexadecimal values. If you want to examine a return value and compare the value with the error codes listed in this documentation, then display the error code in hexadecimal format.

The figure below shows the structure of a system function error code in hexadecimal format.

Error code, e.g. W#16#8081



Event number or error class and single error

If x = ‘0‘, then you are dealing with a specific error code of an instruction. The specific error code is included in the description of the individual instruction.

If x > = ‘0‘, then you are dealing with a general error code of an instruction. In this case x is the number of the instruction parameter that has caused the error. The possible general error codes are listed in the following table.

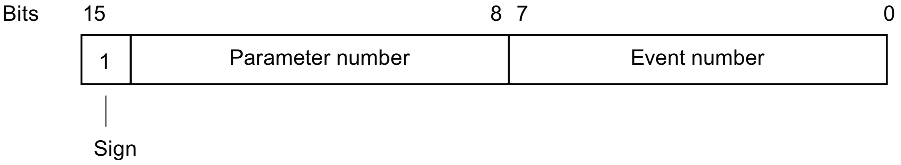
Sign bit = 1 indicates that an error has occurred.

General error information

The general error code indicates errors that can occur in all instructions. A general error code consists of the following two numbers:

- A parameter number from 1 to 111, where 1 indicates the first parameter of the called instruction, 2 the second parameter, and so forth.
- An event number from 0 to 127. The event number indicates that a synchronous error occurred.

The following table lists the codes for general errors and an explanation of each error.



Note

If a general error code was entered in RET_VAL, then the following situations are possible:

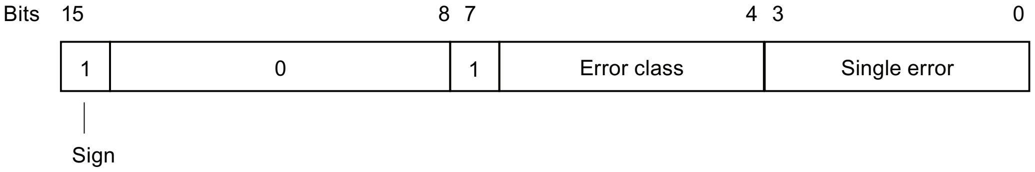
- The action associated with the instruction may have been started or already completed.
- A specific instruction error may have occurred when the action was performed. As a result of a general error that occurred later, the specific error could, however, no longer be indicated.

Specific error information

Some instructions have a return value that provides an error code specific for the instruction. A specific error code indicates errors that can occur only in specific instructions.

A specific error code consists of the following two numbers:

- An error class from 0 to 7.
- An error number from 0 to 15.



General error codes

The following table explains the general error codes of a return value. The error code is shown in hexadecimal format. The letter x in each code number is simply a place holder and represents the number of the system function parameter that caused the error.

General error codes

Error code (W#16#...)	Explanation
8x7F	Internal error This error code indicates an internal error at parameter x.
8x01	Illegal syntax ID at an VARIANT parameter
8x22	Range length error when reading a parameter.
8x23	Range length error when writing a parameter. This error code indicates that the parameter x is located either entirely or partly outside the range of an address, or that the length of a bit range is not a multiple of 8 with an VARIANT parameter.
8x24	Range error when reading a parameter.
8x25	Range error when writing a parameter. This error code indicates that the parameter x is located in a range that is illegal for the system function. Refer to the descriptions of the individual functions for information about the illegal ranges.
8x26	The parameter contains a timer cell number that is too high. This error code indicates that the timer cell specified in parameter x does not exist.
8x27	The parameter contains a counter cell number that is too high (counter number error). This error code indicates that the counter cell specified in parameter x does not exist.
8x28	Alignment error when reading a parameter.
8x29	Alignment error when writing a parameter. This error code indicates that the reference to parameter x is an operand with bit address that is not equal to 0.
8x30	The parameter is located in a read-only global DB.
8x31	The parameter is located in a read-only instance DB. This error code indicates that parameter x is located in a read-only data block. If the data block was opened by the system function itself, the system function always returns the value W#16#8x30.
8x32	The parameter contains a DB number that is too high (DB number error).
8x34	The parameter contains an FC number that is too high (FC number error).
8x35	The parameter contains an FB number that is too high (FB number error). This error code indicates that parameter x contains a block number higher than the highest permitted number.
8x3A	The parameter contains the number of a DB that is not loaded.
8x3C	The parameter contains the number of an FC that is not loaded.
8x3E	The parameter contains the number of an FB that is not loaded.
8x42	An access error occurred while the system was attempting to read a parameter from the peripheral input area.

Error code (W#16#...)	Explanation
8x43	An access error occurred while the system was attempting to write a parameter to the peripheral output area.
8x44	Error in the nth (n > 1) read access after an error occurred.
8x45	Error in the nth (n > 1) write access after an error occurred. This error code indicates that access to the required parameter is denied.

9.8.2 Basic instructions

9.8.2.1 LAD

Bit logic operations

---| |---: Normally open contact

Description

The activation of the normally open contact depends on the signal state of the associated operand. If the signal state of the operand is "1", the normally open contact is closed and the signal state at the output of the instruction is set to "1".

If the signal state of the operand is "0", the normally open contact is not activated and the signal state at the output of the instruction is set to "0".

Two or more normally open contacts are linked bit-by-bit by AND when connected in series. With a series connection, power flows when all contacts are closed.

The normally open contacts are linked by OR when connected in parallel. With a parallel connection, power flows when one of the contacts is closed.

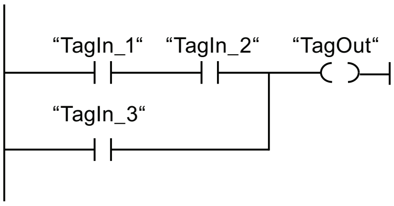
Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L	Operand whose signal state is queried.

Example

The following example shows how the instruction works:



Operand "TagOut" is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have the signal state "1".
- The operand "TagIn_3" has the signal state "1".

See also

Overview of the valid data types (Page 741)

Example of controlling a conveyor belt (Page 1203)

Example of detecting the fill level of a storage area (Page 1206)

Example of controlling room temperature (Page 1209)

---| / |---: Normally closed contact

Description

The activation of the normally closed contact depends on the signal state of the associated operand. If the signal state of the operand is "1", the normally closed contact is opened and the signal state at the output of the instruction is set to "0".

If the signal state of the operand is "0", the normally closed contact is not activated and the signal state at the output of the instruction is set to "1".

Two or more normally closed contacts are linked bit-by-bit by AND when connected in series. With a series connection, power flows when all contacts are closed.

The normally closed contacts are linked by OR when connected in parallel. With a parallel connection, power flows when one of the contacts is closed.

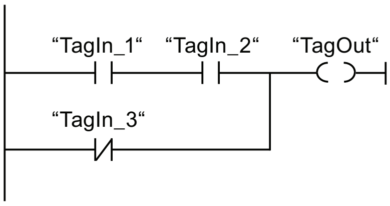
Parameters

The following table shows the parameters of the instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L	Operand whose signal state is queried.

Example

The following example shows how the instruction works:



Operand "TagOut" is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have the signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Overview of the valid data types (Page 741)

Example of detecting the direction of a conveyor belt (Page 1204)

Example of controlling room temperature (Page 1209)

Example of detecting the fill level of a storage area (Page 1206)

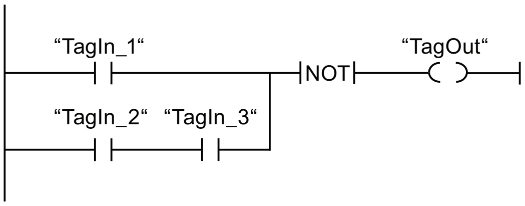
--|NOT|--: Invert RLO

Description

You can use the "Invert RLO" instruction to invert the signal state of the result of logic operation (RLO). If the signal state is "1" at the input of the instruction, the output of the instruction has signal state "0". If the signal state is "0" at the input of the instruction, the output has the signal state "1".

Example

The following example shows how the instruction works:



Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The signal state of the operands "TagIn_2" and "TagIn_3" is "1".

---( )---: Assignment

Description

You can use the "Assignment" instruction to set the bit of a specified operand. If the result of logic operation (RLO) at the input of the coil has signal state "1", the specified operand is set to signal state "1". If the signal state is "0" at the input of the coil, the bit of the specified operand is reset to "0".

The instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output.

The "Assignment" instruction can be placed at any position in the network.

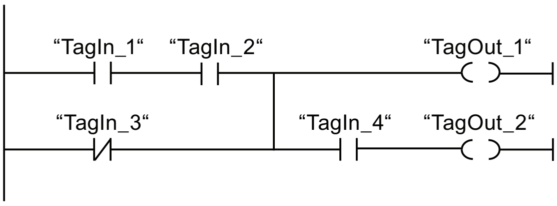
Parameter

The following table shows the parameters of the "Assignment" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:



The "TagOut_1" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

The "TagOut_2" operand is set when one of the following conditions is fulfilled:

- Operands "TagIn_1", "TagIn_2", and "TagIn_4" have signal state "1".
- The signal state of the "TagIn_3" operand is "0" and the signal state of the "TagIn_4" operand is "1".

See also

Overview of the valid data types (Page 741)

Example of detecting the fill level of a storage area (Page 1206)

Example of controlling room temperature (Page 1209)

--(/)--: Negate assignment

Description

The "Negate assignment" instruction inverts the result of logic operation (RLO) and assigns it to the specified operand. When the RLO at the input of the coil is "1", the operand is reset. When the RLO at the input of the coil is "0", the operand is set to signal state "1".

The instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output of the coil.

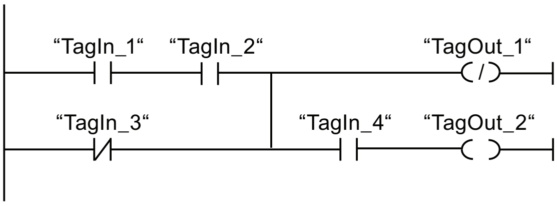
Parameters

The following table shows the parameters of the "Negate assignment" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:



Operand "TagOut_1" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have the signal state "1".
- The signal state of the operand "TagIn_3" is "0".

See also

Overview of the valid data types (Page 741)

---(R)---: Reset output

Description

You can use the "Reset output" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO = "1"), the specified operand is reset to "0". If the RLO at the input of the coil is "0" (no signal flow to the coil), the signal state of the specified operand remains unchanged.

Executing the instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output of the coil.

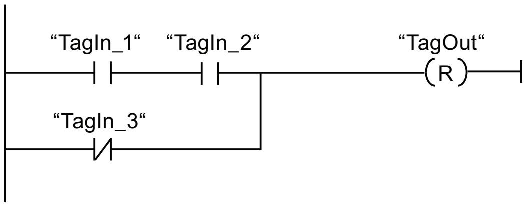
Parameter

The following table shows the parameters of the "Reset output" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand which is reset with RLO = "1".

Example

The following example shows how the instruction works:



Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

See also

Overview of the valid data types (Page 741)

Example of controlling a conveyor belt (Page 1203)

Example of detecting the direction of a conveyor belt (Page 1204)

---(S)---: Set output

Description

You can use the "Set output" instruction to set the signal state of a specified operand to "1".

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO = "1"), the specified operand is set to "1". If the RLO at the input of the coil is "0" (no signal flow to the coil), the signal state of the specified operand remains unchanged.

Executing the instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output of the coil.

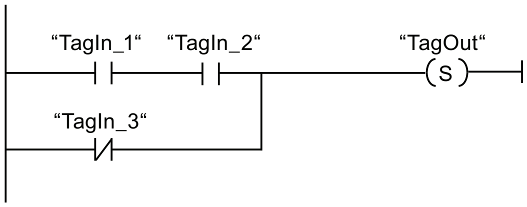
Parameter

The following table shows the parameters of the "Set output" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand which is set with RLO = "1".

Example

The following example shows how the instruction works:



The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

See also

Overview of the valid data types (Page 741)

Example of controlling a conveyor belt (Page 1203)

Example of detecting the direction of a conveyor belt (Page 1204)

SET_BF: Set bit field

Description

You use the "Set bit field" instruction to set several bits starting from a certain address.

You determine the number of bits to be set using the value of <Operand1>. The address of the first bit to be set is defined by <Operand2>. If the value of <Operand1> is greater than the number of bits in a selected byte, then the bits of the next byte will be set. The bits remain set until they are explicitly reset, for example, by another instruction.

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If the RLO at the input of the coil is "0", the instruction does not execute.

The "Set bit field" instruction can also be placed without preceding logic operation at the start or end of the current path.

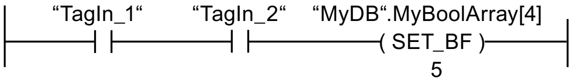
Parameter

The following table shows the parameters of the "Set bit field" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	UINT	Constant	Number of bits to be set
<Operand2>	Output	BOOL	I, Q, M With a DB or an IDB, an element of a array [..] of BOOL	Pointer to the first bit to be set.

Example

The following example shows the mode of operation of the "Set bit field" instruction:



If the operands "TagIn_1" and "TagIn_2" have the signal state "1", 5 bits are set starting at the address of the operand "MyDB".MyBoolArray[4].

See also

Overview of the valid data types (Page 741)

RESET_BF: Reset bit field

Description

You use the "Reset bit field" instruction to re set several bits starting from a certain address.

You specify the number of bits to be reset using the value of <Operand1>. The address of the first bit to be reset is specified by <Operand2>. If the value of <Operand1> is greater than the number of bits in a selected byte, the bits of the next byte will be reset. The bits remain set until they are explicitly reset, for example, by another instruction.

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If the RLO at the input of the coil is "0", the instruction does not execute.

The "Reset bit field" instruction can also be placed without preceding logic operation at the start or end of the current path.

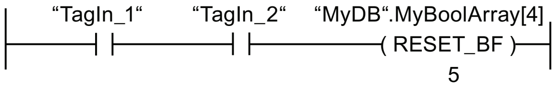
Parameters

The following table shows the parameters of the "Reset bit field" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	UINT	Constant	Number of bits to be reset
<Operand2>	Output	BOOL	I, Q, M With a DB or an IDB, an element of a array [..] of BOOL	Pointer to the first bit to be reset.

Example

The following example shows the mode of operation of the "Reset bit field" instruction:



If the operands "TagIn_1" and "TagIn_2" have the signal state "1", 5 bits are reset starting at the address of the operand "MyDB".MyBoolArray[4].

See also

Overview of the valid data types (Page 741)

SR: Set/reset flip-flop

Description

You can use the "Set/reset flip-flop" instruction to set or reset the bit of a specified operand based on the signal state of the inputs S and R1. If the signal state is "1" at input S and "0" at input R1, the specified operand is set to "1". If the signal state is "0" at input S and "1" at input R1, the specified operand will be reset to "0".

Input R1 takes priority over input S. When the signal state is "1" at both inputs S and R1, the signal state of the specified operand is reset to "0".

The instruction is not executed if the signal state at the two inputs S and R1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

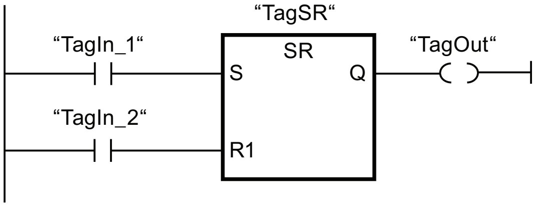
Parameter

The following table shows the parameters of the "Set/reset flip-flop" instruction:

Parameter	Declaration	Data type	Memory area	Description
S	Input	BOOL	I, Q, M, D, L	Enable setting
R1	Input	BOOL	I, Q, M, D, L	Enable resetting
<Operand>	Output	BOOL	I, Q, M, D, L	Operand that is set or reset
Q	Output	BOOL	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the "Set/reset flip-flop" instruction works:



The operands "TagSR" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagSR" and "TagOut" are reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 741)

RS: Reset/set flip-flop

Description

You can use the "Reset/set flip-flop" instruction to reset or set the bit of a specified operand based on the signal state of the inputs R and S1. If the signal state is "1" at input R and "0" at input S1, the specified operand will be reset to "0". If the signal state is "0" at input R and "1" at input S1, the specified operand is set to "1".

Input S1 takes priority over input R. When the signal state is "1" at both inputs R and S1, the signal state of the specified operand is set to "1".

The instruction is not executed if the signal state at the two inputs R and S1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

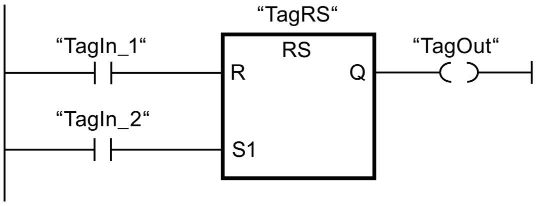
Parameter

The following table shows the parameters of the "Reset/set flip-flop" instruction:

Parameter	Declaration	Data type	Memory area	Description
R	Input	BOOL	I, Q, M, D, L	Enable resetting
S1	Input	BOOL	I, Q, M, D, L	Enable setting
<Operand>	Output	BOOL	I, Q, M, D, L	Operand that is reset or set
Q	Output	BOOL	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the "Reset/set flip-flop" instruction works:



The operands "TagRS" and "TagOut" are reset when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagRS" and "TagOut" are set when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

--|P|--: Scan operand for positive signal edge

Description

You can use the "Scan operand for positive signal edge" instruction to determine whether there is a "0" to "1" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan stored in an edge memory bit (<operand2>). If the instruction detects a change in the result of logic operation from "0" to "1", there is a positive, rising edge.

If a rising edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

You specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. You specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit would be overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

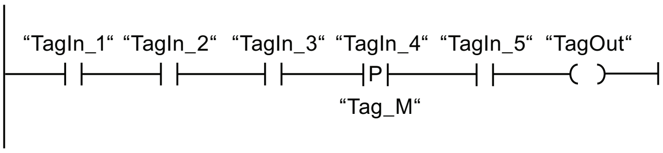
Parameter

The following table shows the parameters of the "Scan operand for positive signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	BOOL	I, Q, M, D, L	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit in which the signal state of the previous scan is saved

Example

The following example shows how the "Scan operand for positive signal edge" instruction works:



Operand "TagOut" is set when the following conditions are fulfilled:

- Operands "TagIn_1", "TagIn_2", and "TagIn_3" have signal state "1".
- There is a rising edge at operand "TagIn_4". The signal state of the previous scan is stored in the edge memory bit "Tag_M".
- The signal state of the operand "TagIn_5" is "1".

See also

Overview of the valid data types (Page 741)

Example of detecting the direction of a conveyor belt (Page 1204)

--|N|--: Scan operand for negative signal edge

Description

You can use the "Scan operand for negative signal edge" instruction to detect whether there is a change from "1" to "0" in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan that is saved in an edge memory bit <Operand2>. If the instruction detects a change in the result of logic operation from "1" to "0", there is a negative, falling edge.

If a falling edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

You specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. You specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit would be overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

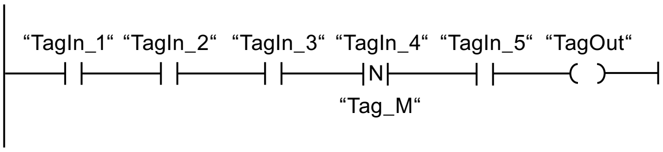
Parameter

The following table shows the parameters of the "Scan operand for negative signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	BOOL	I, Q, M, D, L	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit in which the signal state of the previous scan is saved

Example

The following example shows how the "Scan operand for negative signal edge" instruction works:



Operand "TagOut" is set when the following conditions are fulfilled:

- Operands "TagIn_1", "TagIn_2", and "TagIn_3" have signal state "1".
- There is a falling edge at operand "TagIn_4". The signal state of the previous scan is stored in the edge memory bit "Tag_M".
- The signal state of the operand "TagIn_5" is "1".

See also

Overview of the valid data types (Page 741)

--(P)--: Set operand on positive signal edge

Description

You can use the "Set operand on positive signal edge" instruction to set a specified operand (<Operand1>) when there is a "0" to "1" transition in the power flow. The instruction compares the current result of logic operation (RLO) with the result of logic operation from the previous query, which is saved in an edge memory bit (<Operand2>). If the instruction detects a change in the power flow from "0" to "1", there is a positive, rising edge.

When there is a positive edge, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

Specify the operand to be set (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit has to lie in a DB (static area for FB) or in the bit memory area.

The instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output of the coil.

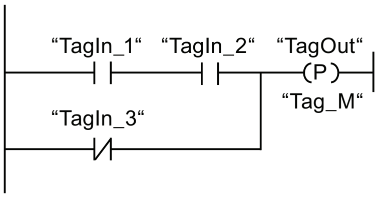
Parameters

The following table shows the parameters of the "Set operand on positive signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Output	BOOL	I, Q, M, D, L	Operand which is set by a positive edge.
<Operand2>	InOut	BOOL	I, Q, M, D	Edge memory bit

Example

The following example shows how the instruction works:



Operand "TagOut" is set for one program cycle, when the signal state at the input of the coil switches from "0" to "1" (positive edge). In all other cases, the operand "TagOut" has the signal state "0".

See also

Overview of the valid data types (Page 741)

--(N)--: Set operand on negative signal edge

Description

You can use the "Set operand on negative signal edge" instruction to set a specified operand (<Operand1>) when a "1" to "0" change is detected in the power flow. The instruction compares the current result of logic operation (RLO) with the result of logic operation from the previous query, which is saved in an edge memory bit (<Operand2>). If the instruction detects a change in the power flow from "1" to "0", there is a negative, falling edge.

When a negative edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

Specify the operand to be set (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit has to lie in a DB (static area for FB) or in the bit memory area.

The instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output of the coil.

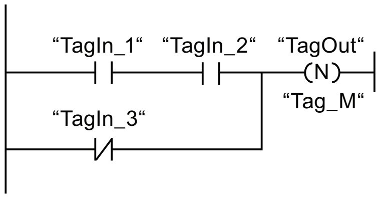
Parameters

The following table shows the parameters of the "Set operand on negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Output	BOOL	I, Q, M, D, L	Operand which is set by a negative edge.
<Operand2>	InOut	BOOL	I, Q, M, D	Edge memory bit

Example

The following example shows how the instruction works:



Operand "TagOut" is set for one program cycle, when the signal state at the input of the coil switches from "1" to "0" (negative edge). In all other cases, the operand "TagOut" has the signal state "0".

See also

Overview of the valid data types (Page 741)

P_TRIG: Scan RLO for positive signal edge

Description

Use the "Scan RLO for positive signal edge" instruction to query a "0" to "1" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the result of logic operation (RLO) with the signal state of the previous query, which is saved in an edge memory bit (<Operand>). If the instruction detects a change in the RLO from "0" to "1", there is a positive, rising edge.

If a rising edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area with an FB) or in the bit memory area.

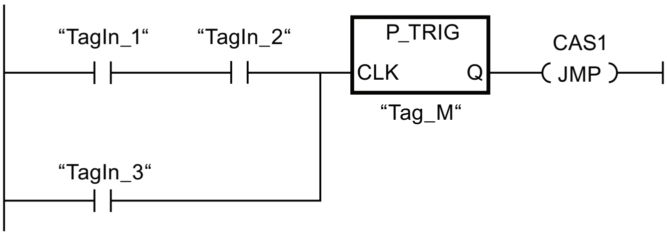
Parameters

The following table shows the parameters of the "Scan RLO for positive signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	I, Q, M, D, L	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the previous query is saved in the edge memory bit "Tag_M". If a "0" to "1" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 741)

N_TRIG: Scan RLO for negative signal edge

Description

Use the "Scan RLO for negative signal edge" instruction to query a "1" to "0" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the result of logic operation with the signal state of the previous query, which is saved in an edge memory bit (<Operand>). If the instruction detects a change in the RLO from "1" to "0", there is a negative, falling edge.

If a falling edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area with an FB) or in the bit memory area.

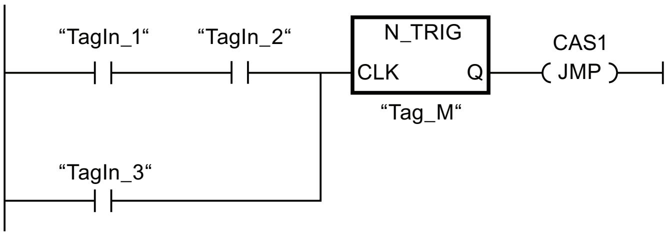
Parameters

The following table shows the parameters of the "Scan RLO for negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	I, Q, M, D, L	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the previous query is saved in the edge memory bit "Tag_M". If there is a change in the signal state of the RLO from "1" to "0", the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 741)

Timer operations

TP: Generate pulse

Description

The instruction "Generate pulse" sets output Q for duration PT. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. Output Q is set for the duration PT, regardless of the subsequent course of the input signal. Even if a new positive signal edge is detected, the signal state at the output Q is not affected as long as the PT time is running.

The current time value can be queried at the ET output. The time value starts at T#0s and ends when the value of duration PT is reached. If duration PT is reached and the signal state at input IN is "0", the ET output is reset.

Each call of the "Generate pulse" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER or TP that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

The execution of the "Generate pulse" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

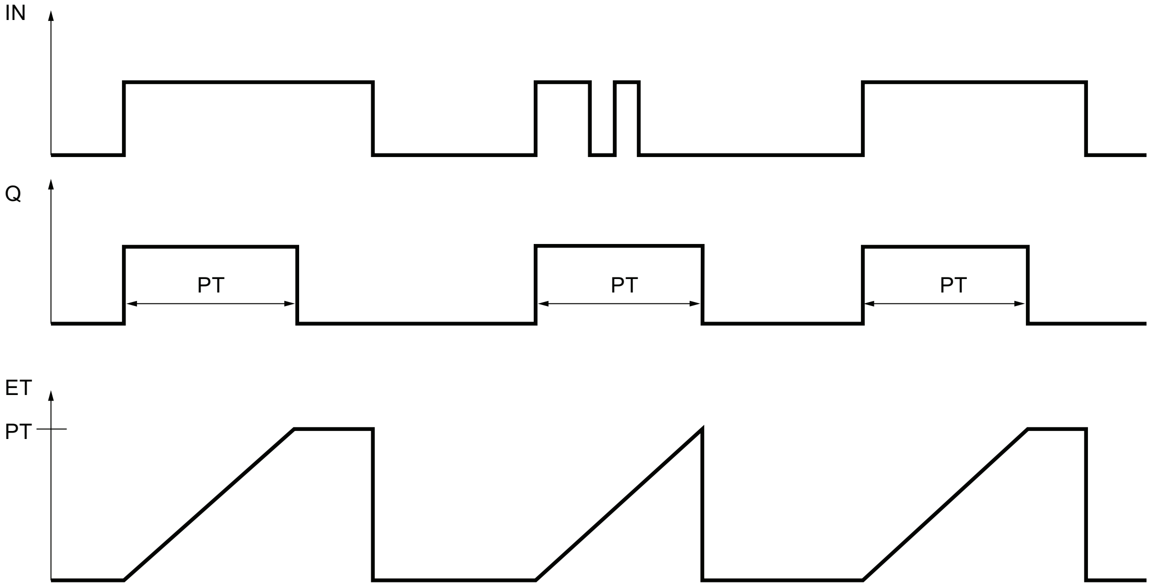
The following table shows the parameters of the "Generate pulse" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L or constant	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Pulse output
ET	Output	TIME	I, Q, M, D, L	Current time value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate pulse" instruction:



See also

Overview of the valid data types (Page 741)

Example of controlling room temperature (Page 1209)

TON: Generate on-delay

Description

The instruction "Generate on-delay" delays setting of the output Q by the programmed duration PT. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the duration PT expires, the output Q has the signal state "1". Output Q remains set as long as the start input is still "1". When the signal state at the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new positive signal edge is detected at the start input.

The current time value can be queried at the ET output. The time value starts at T#0s and ends when the value of duration PT is reached. The ET output is reset as soon as the signal state at the IN input changes to "0".

Each call of the "Generate on-delay" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER or TON that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

The execution of the "Generate on-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

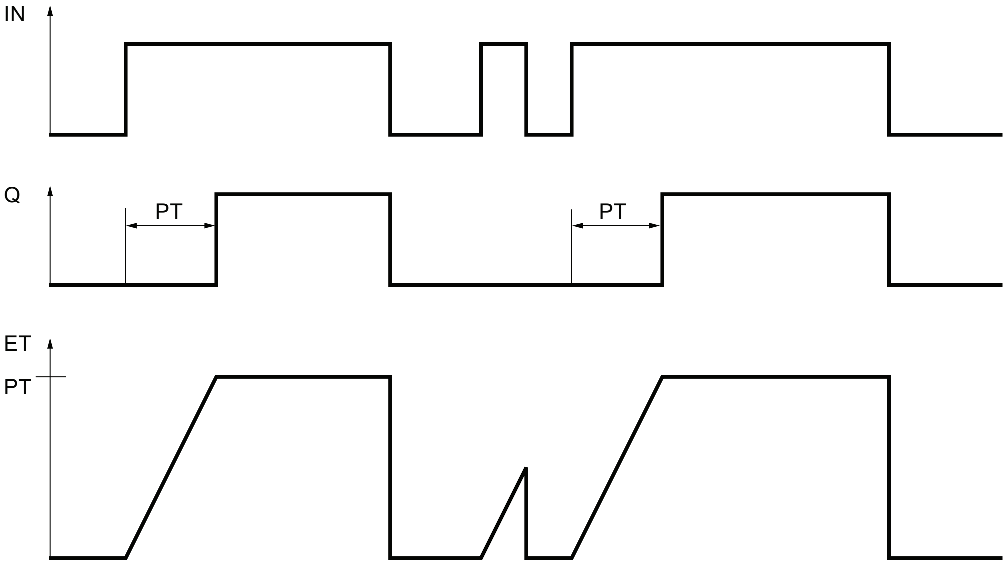
The following table shows the parameters of the "Generate on-delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L or constant	Duration of the on delay. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is set when the time PT expires.
ET	Output	TIME	I, Q, M, D, L	Current time value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate on-delay" instruction:



See also

Overview of the valid data types (Page 741)

TOF: Generate off-delay

Description

The instruction "Generate off-delay" delays resetting of the output Q by the programmed duration PT. The Q output is set when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). When the signal state at input IN changes back to "0", the programmed time PT starts. Output Q remains set as long as the duration PT is running. When duration PT expires, the Q output is reset. If the signal state at the IN input changes to "1" before the duration PT expires, the time is reset. The signal state at the output Q will continue to be "1".

The current time value can be queried at the ET output. The time value starts at T#0s and ends when the value of duration PT is reached. When the duration PT expires, the ET output remains set to the current value until input IN changes back to "1". If input IN switches to "1" before the duration PT has expired, the ET output is reset to the value T#0s.

Each call of the "Generate off-delay" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER or TOF that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

The execution of the "Generate off-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

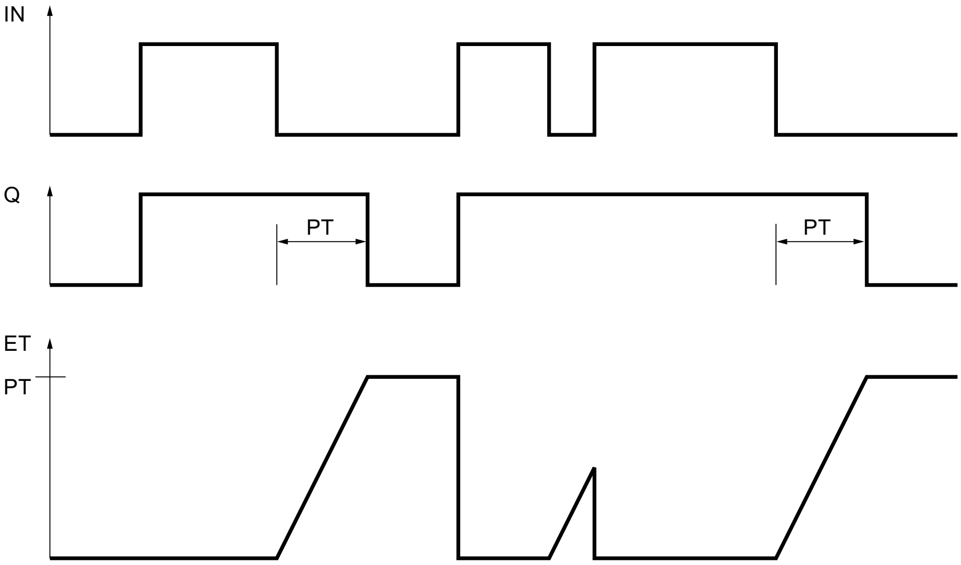
The following table shows the parameters of the "Generate off-delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L or constant	Duration of the off delay. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is reset when the timer PT expires.
ET	Output	TIME	I, Q, M, D, L	Current time value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate off-delay" instruction:



See also

Overview of the valid data types (Page 741)

TONR: Time accumulator

Description

The "Time accumulator" instruction accumulates time values within a period set by parameter PT. When the signal state at input IN changes from "0" to "1" (positive signal edge), the instruction executes and the duration PT starts. While the duration PT is running, the timer values are accumulated that are recorded when the IN input has signal state "1". The accumulated time is written to output ET and can be queried there. When the duration PT expires, the output Q has the signal state "1". The Q parameter remains set to "1", even when the signal state at the IN parameter changes from "1" to "0" (negative signal edge).

The R input resets the outputs ET and Q regardless of the signal state at the start input.

Each call of the "Time accumulator" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER or TONR that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

The execution of the "Time accumulator" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

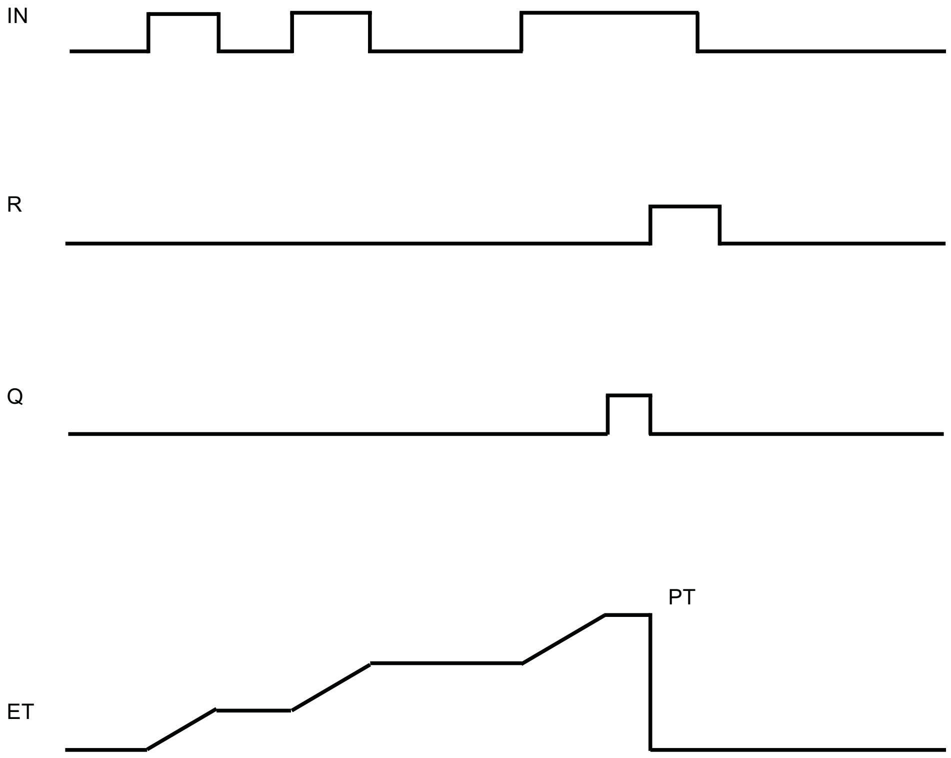
The following table shows the parameters of the "Time accumulator" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
R	Input	BOOL	I, Q, M, D, L	Reset input
PT	Input	TIME	I, Q, M, D, L or constant	Maximum duration of time recording The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is set when the time PT expires.
ET	Output	TIME	I, Q, M, D, L	Accumulated time

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Time accumulator" instruction:



See also

Overview of the valid data types (Page 741)

---(TP)---: Start pulse timer

Description

The "Start pulse timer" instruction starts an IEC timer with a specified duration as pulse. The IEC timer is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The IEC timer runs for the specified time regardless of any subsequent changes in the RLO. The run of the IEC timer is also not affected by the detection of a new positive signal edge. As long as the IEC timer is running, the querying of the timer status for "1" returns the signal state "1". When the IEC timer has expired, the timer status returns the signal state "0".

The "Start pulse timer" instruction stores its data in a structure of the data type IEC_TIMER or TP. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

The current timer status is stored in the Q structure component of the IEC timer. You can use a normally open contact to query timer status for "1" or a normally closed contact for "0".

The execution of the "Start pulse timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

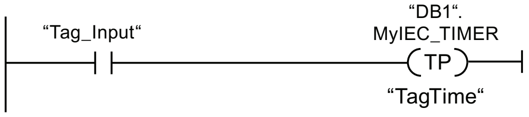
The following table shows the parameters of the instruction "Start pulse timer":

Parameters	Declaration	Data type	Memory area	Description
<Time duration>	Input	TIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs
<IEC timer>	InOut	IEC_TIMER/TP	D, L	IEC timer that is started

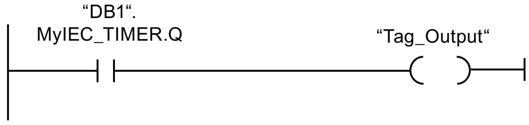
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Start pulse timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "0" to "1". Timer "DB1."MyIEC_TIMER starts running for the time duration that is stored in operand "TagTime".



As long as the timer "DB1".MyIEC_TIMER is running, the timer status ("DB1"MyIEC_TIMER.Q) has the signal state "1" and the operand "Tag_Output" is set. When the IEC timer has expired, the signal state of the timer status changes back to "0" and the "Tag_Output" operand is reset.

See also

Overview of the valid data types (Page 741)

---(TON)---: Start on-delay timer

Description

The "Start on-delay timer" instruction starts an IEC timer with a specified duration as on-delay timer. The IEC timer is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The IEC timer runs for the specified time. A query of the timer status for "1" returns signal state "1" if the time has expired and the RLO at the input of the instruction is "1". If the RLO changes to "0" before the time expires, the IEC timer is reset. In this case, querying the timer status for "1" returns signal state "0". The IEC timer restarts when the next positive signal edge is detected at the input of the instruction.

The "Start on-delay timer" instruction stores its data in a structure of the data type IEC_TIMER or TON. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

The current timer status is stored in the Q structure component of the IEC timer. You can use a normally open contact to query timer status for "1" or a normally closed contact for "0".

The execution of the "Start on-delay timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

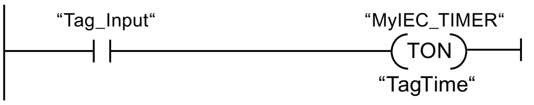
The following table shows the parameters of the instruction "Start on-delay timer":

Parameters	Declaration	Data type	Memory area	Description
<Time duration>	Input	TIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs
<IEC timer>	InOut	IEC_TIMER/TON	D, L	IEC timer that is started

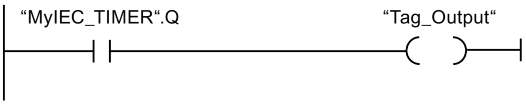
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Start on-delay timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "0" to "1". Timer "MyIEC_TIMER" starts running for the time duration that is stored in operand "TagTime".



If the timer "MyIEC_TIMER" has expired and the operand "Tag_Input" has the signal state "1", querying the timer status ("MyIEC_TIMER".Q) returns signal state "1" and the "Tag_Output" operand is set. When the signal state of the operand "Tag_Input" changes to "0", the querying of the timer status returns the signal state "0" and the operand "Tag_Output" is reset.

See also

Overview of the valid data types (Page 741)

---(TOF)---: Start off-delay timer

Description

The "Start off-delay timer" instruction starts an IEC timer with a specified duration as off-delay timer. The query of the timer status for "1" returns the signal state "1" if the result of the logic operation (RLO) at the input of the instruction has the signal state "1". When the RLO changes from "1" to "0" (negative signal edge), the IEC timer starts with the specified time. The timer status remains at signal state "1" as long as the IEC timer is running. When the timer has run out and the RLO at the input of the instruction has the signal state "0", the timer status is set to the signal state "0". If the RLO changes to "1" before the time expires, the IEC timer is reset and the timer status remains at signal state "1".

The "Start off-delay timer" instruction stores its data in a structure of the data type IEC_TIMER or TOF. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

The current timer status is stored in the Q structure component of the IEC timer. You can use a normally open contact to query timer status for "1" or a normally closed contact for "0".

The execution of the "Start off-delay timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

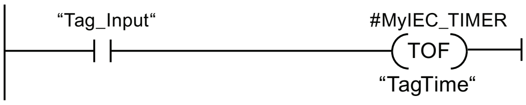
The following table shows the parameters of the instruction "Start off-delay timer":

Parameters	Declaration	Data type	Memory area	Description
<Time duration>	Input	TIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs
<IEC timer>	InOut	IEC_TIMER/TOF	D, L	IEC timer that is started

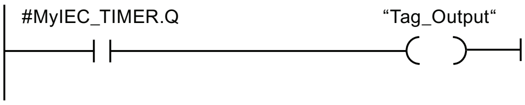
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Start off-delay timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "1" to "0". Timer #MyIEC_TIMER starts running for the time duration that is stored in operand "TagTime".



As long as timer #MyIEC_TIMER is running, the query of the timer status (#MyIEC_TIMER.Q) returns the signal state "1" and operand "Tag_Output" is set. If the timer has expired and the operand "Tag_Input" has the signal state "0", the query of the timer status returns the signal state "0". If the signal state of the operand "Tag_Input" changes to "1" before timer #MyIEC_TIMER expires, the timer is reset. When the signal state of the operand "Tag_Input" is "1", the query of the timer status returns the signal state "1".

See also

Overview of the valid data types (Page 741)

---(TONR)---: Time accumulator

Description

The "Time accumulator" instruction records how long the signal is at the input of instruction "1". The instruction is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The time is recorded as long at the RLO is "1". If the RLO changes to "0", the instruction is halted. If the RLO changes back to "1", the time recording is continued. The query of the timer status for "1" returns the signal state "1" if the recorded time exceeds the value of the specified duration and the RLO at the input of coil is "1".

The timer status and the currently expired timer can be reset to "0" using the "Reset timer" instruction.

The "Time accumulator" instruction stores its data in a structure of the data type IEC_TIMER or TONR. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

The current timer status is stored in the Q structure component of the IEC timer. You can use a normally open contact to query timer status for "1" or a normally closed contact for "0".

The execution of the "Time accumulator" instruction requires a preceding logic operation. It can be placed only at the end of the network.

Parameters

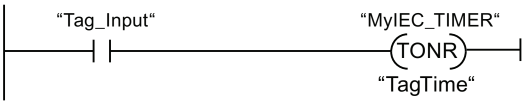
The following table shows the parameters of the "Time accumulator" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Time duration>	Input	TIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs
<IEC timer>	InOut	IEC_TIMER/TONR	D, L	IEC timer that is started

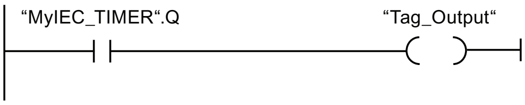
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Time accumulator" instruction executes on a positive signal edge in the RLO. The time is recorded as long as the operand "Tag_Input" has the signal state "1".



If the recorded time exceeds the value of the operand "TagTime", the query of the timer status ("MyIEC_TIMER".Q) returns the signal state "1" and the operand "Tag_Output" is set.

See also

Overview of the valid data types (Page 741)

---(RT)---: Reset timer

Description

The "Reset timer" instruction resets an IEC timer to "0". The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If current is flowing to the coil (RLO is "1"), the structure components of the timer in the specified data block are reset to "0". If the RLO at the input of the instruction is "0", the timer remains unchanged.

The instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output of the coil.

You assign the "Reset timer" instruction an IEC timer that has been declared in the program.

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

Parameters

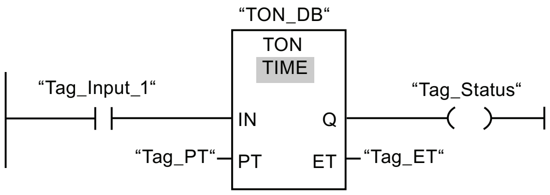
The following table shows the parameters of the instruction "Reset timer":

Parameters	Declaration	Data type	Memory area	Description
<IEC timer>	Output	IEC_TIMER, TON, TOF, TP	D, L	IEC timer that is reset.

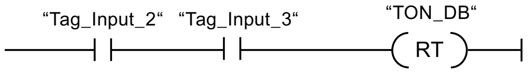
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Generate on-delay" instruction executes when the signal state of the "Tag_Input_1" operand changes from "0" to "1". The timer stored in the "TON_DB" instance data block starts running for the time duration specified by operand "Tag_PT".



If operands "Tag_Input_2" and "Tag_Input_3" have the signal state "1", the "Reset timer" instruction is executed and the timer stored in the "TON_DB" data block is reset.

See also

Overview of the valid data types (Page 741)

---(PT)---: Load time duration

Description

The instruction "Load time duration" sets the duration of an IEC timer. The instruction is executed in every cycle when the result of logic operation (RLO) at the input of the instruction has the signal state "1". The instruction writes the specified time to the structure of the specified IEC timer.

Note

If the specified IEC timer is running while the instruction executes, the instruction overwrites the current time of the specified IEC timer. This can change the timer status of the IEC timer.

You assign an IEC timer declared in the program to the "Load time duration" instruction.

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

Parameters

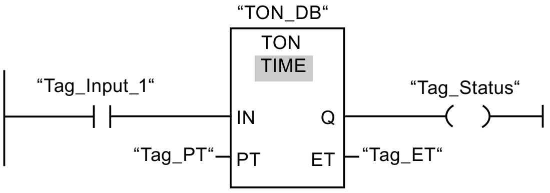
The following table shows the parameters of the "Load time duration" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Time duration>	Input	TIME	I, Q, M, D, L or constant	Time duration that is set
<IEC timer>	Output	IEC_TIMER, TON, TOF, TP	D, L	IEC timer whose duration is set

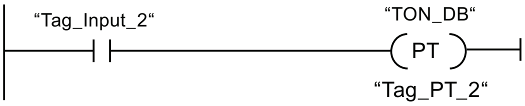
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Generate on-delay" instruction executes when the signal state of the "Tag_Input_1" operand changes from "0" to "1". The IEC timer stored in the "TON_DB" instance data block starts running for the time duration specified by operand "Tag_PT".



When the "Tag_Input_2" operand has signal state "1", the "Load time duration" instruction executes. The instruction writes the time duration "Tag_PT_2" to the "TON_DB" instance data block, overwriting the value of the "Tag_PT" operand in the data block. The signal state of the timer status may therefore change at the next query.

See also

Overview of the valid data types (Page 741)

Counter operations

CTU: Count up

Description

The instruction "Count up" counts up the value at output CV. When the signal state at the CU input changes from "0" to "1" (positive signal edge), the instruction executes and the current count value at the CV output is incremented by one. When the instruction executes for the first time, the current count value at the CV output is set to zero. The count value is incremented each time a positive signal edge is detected, until it reaches the high limit for the data type specified at the CV output. When the high limit is reached, the signal state at the CU input no longer has an effect on the instruction.

You can scan the counter status at the Q output. The signal state at the Q output is decided by the parameter PV. If the current count value is greater than or equal to the value of the PV parameter, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0". You can also specify a constant for the PV parameter.

The value at the CV output is reset to zero when the signal state at the R input changes to "1". As long as the R input has signal state "1", the signal state at the CU input has no effect on the instruction.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

- Data block of system data type IEC_counter (global DB):
  - IEC_SCOUNTER / IEC_USCOUNTER
  - IEC_COUNTER / IEC_UCOUNTER
  - IEC_DCOUNTER / IEC_UDCOUNTER
- Local tag:
  - CTU_SINT / CTU_USINT
  - CTU_INT / CTU_UINT
  - CTU_DINT / CTU_UDINT

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_COUNTER (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTU in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The execution of the "Count up" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count up" instruction:

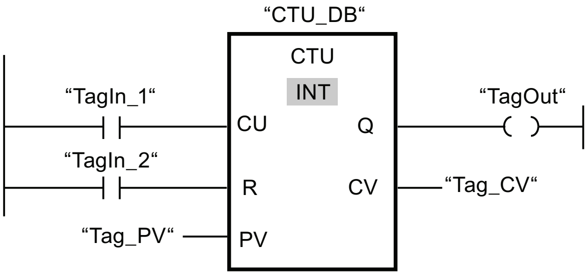
Parameters	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count input
R	Input	BOOL	I, Q, M, D, L	Reset input
PV	Input	Integers	I, Q, M, D, L or constant	Value at which the output Q is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	Integers	I, Q, M, D, L	Current count value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count up" instruction executes and the current count value of the "Tag_CV" operand is incremented by one. With each additional positive signal edge, the count value is incremented until the high limit of the specified data type (32 767) is reached.

The value of the PV parameter is adopted as the limit for determining the "TagOut" output. The "TagOut" output has signal state "1" as long as the current count value is greater than or equal to the value of the "Tag_PV" operand. In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 741)

CTD: Count down

Description

You can use the "Count down" instruction to decrement the value at output CV. When the signal state at the CD input changes from "0" to "1" (positive signal edge), the instruction executes and the current count value at the CV output is decremented by one. When the instruction executes the first time, the count value of the CV parameter is set to the value of the PV parameter. Each time a positive signal edge is detected, the count value is decremented until it reaches the low limit value of the specified data type. When the low limit is reached, the signal state at the CD input no longer has an effect on the instruction.

You can scan the counter status at the Q output. If the current count value is less than or equal to zero, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0". You can also specify a constant for the PV parameter.

The value at the CV output is set to the value of the PV parameter when the signal state at the LD input changes to "1". As long as the LD input has signal state "1", the signal state at the CD input has no effect on the instruction.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

- Data block of system data type IEC_counter (global DB):
  - IEC_SCOUNTER / IEC_USCOUNTER
  - IEC_COUNTER / IEC_UCOUNTER
  - IEC_DCOUNTER / IEC_UDCOUNTER
- Local tag:
  - CTU_SINT / CTU_USINT
  - CTU_INT / CTU_UINT
  - CTU_DINT / CTU_UDINT

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_COUNTER (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTD in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The execution of the "Count down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count down" instruction:

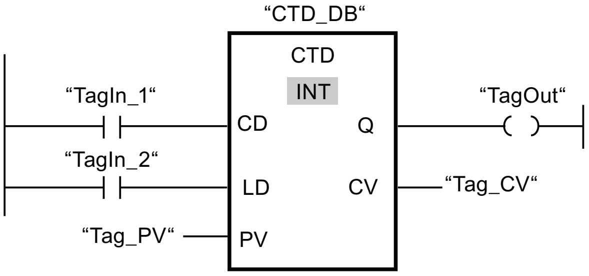
Parameters	Declaration	Data type	Memory area	Description
CD	Input	BOOL	I, Q, M, D, L	Count input
LD	Input	BOOL	I, Q, M, D, L	Load input
PV	Input	Integers	I, Q, M, D, L or constant	Value at which the output Q is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	Integers	I, Q, M, D, L	Current count value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count down" instruction executes and the value at the "Tag_CV" output is decremented by one. With each additional positive signal edge, the count value is decremented until the low limit of the specified data type (-32 768) is reached.

The "TagOut" output has signal state "1" as long as the current count value is less than or equal to zero. In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 741)

CTUD: Count up and down

Description

You can use the "Count up and down" instruction to increment and decrement the count value at the CV output. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current count value is incremented by one and stored at the CV output. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the count value at the CV output is decremented by one. If there is a positive signal edge at the CU and CD inputs in one program cycle, the current count value at the CV output remains unchanged.

The count value can be incremented until it reaches the high limit of the data type specified at the CV output. When the high limit value is reached, the count value is no longer incremented on a positive signal edge. When the low limit of the specified data type is reached, the count value is not decremented any further.

When the signal state at the LD input changes to "1", the count value at the CV output is set to the value of the PV parameter. As long as the LD input has the signal state "1", the signal state at the CU and CD inputs has no effect on the instruction.

The count value is set to zero when the signal state at the R input changes to "1". As long as the R input has signal state "1", a change in the the signal state of the CU, CD and LD inputs has no effect on the "Count up and down" instruction.

You can scan the current status of the up counter at the QU output. If the current count value is greater than or equal to the value of the PV parameter, the QU output is set to signal state "1". In all other cases, the QU output has signal state "0". You can also specify a constant for the PV parameter.

You can scan the current status of the down counter at the QD output. If the current count value is less than or equal to zero, the QD output is set to signal state "1". In all other cases, the QD output has signal state "0".

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

- Data block of system data type IEC_counter (global DB):
  - IEC_SCOUNTER / IEC_USCOUNTER
  - IEC_COUNTER / IEC_UCOUNTER
  - IEC_DCOUNTER / IEC_UDCOUNTER
- Local tag:
  - CTU_SINT / CTU_USINT
  - CTU_INT / CTU_UINT
  - CTU_DINT / CTU_UDINT

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_COUNTER (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTUD in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The execution of the "Count up and down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count up and down" instruction:

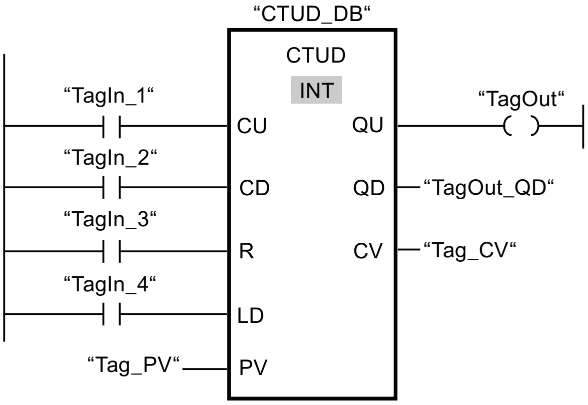
Parameters	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L	Count down input
R	Input	BOOL	I, Q, M, D, L	Reset input
LD	Input	BOOL	I, Q, M, D, L	Load input
PV	Input	Integers	I, Q, M, D, L or constant	Value at which the output QU / QD is set.
QU	Output	BOOL	I, Q, M, D, L	Status of the counter up
QD	Output	BOOL	I, Q, M, D, L	Status of the counter down
CV	Output	Integers	I, Q, M, D, L	Current count value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the signal state at the "TagIn_1" or "TagIn_2" input changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When there is a positive signal edge at the "TagIn_1" input, the current count value is incremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the "TagIn_2" input, the count value is decremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the CU input, the count value is incremented until it reaches the high limit of 32 767. If input CD has a positive signal edge, the count value is decremented until it reaches the low limit value of -32 768.

The "TagOut" output has signal state "1" as long as the current count value is greater than or equal to the value at the "Tag_PV" input. In all other cases, the "TagOut" output has signal state "0".

The "TagOut_QD" output has signal state "1" as long as the current count value is less than or equal to zero. In all other cases, the "TagOut_QD" output has signal state "0".

See also

Example of detecting the fill level of a storage area (Page 1214)

Overview of the valid data types (Page 741)

Example of detecting the fill level of a storage area (Page 1206)

Comparator operations

CMP ==: Equal

Description

You can use the "Equal" instruction to determine if a first comparison value (<Operand1>) is equal to a second comparison value (<Operand2>).

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire current path as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) in the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) in the operand placeholder below the instruction.

The individual characters are compared by means of their ASCII code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character that is different decides the result of the comparison.

The following table shows examples of string comparisons:

<Operand1>	<Operand2>	RLO of instruction
'AA'	'AA'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0

You can also use the "Equal" instruction to compare individual characters of a string (STRING). The number of the character to be compared is given in brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

When IEC Check is selected, the operands to be compared must have the same data type. If IEC Check is not selected, the width (length) of the operands must be the same. When floating-point numbers are compared, the operands to be compared must have the same data type regardless of the setting for the IEC Check.

Parameters

The following table shows the parameters of the "Equal" instruction:

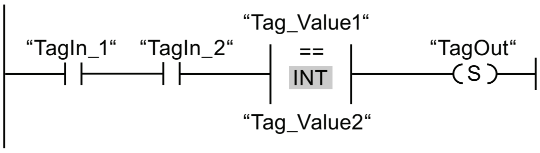
Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers, floating-point numbers, characters, TIME, DATE, TOD, DTL	I, Q, M, D, L or constant	First value to compare
<Operand2>	Input	Bit strings, integers, floating-point numbers, characters, TIME, DATE, TOD, DTL	I, Q, M, D, L or constant	Second value to compare

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have the signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" = "Tag_Value2").

See also

Overview of the valid data types (Page 741)

CMP <>: Not equal

Description

You can use the "Not equal" instruction to determine if a first comparison value (<Operand1>) is unequal to a second comparison value (<Operand2>).

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire current path as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) in the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) in the operand placeholder below the instruction.

The individual characters are compared by means of their ASCII code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character that is different decides the result of the comparison.

The following table shows examples of string comparisons:

<Operand1>	<Operand2>	RLO of instruction
'AA'	'aa'	1
'Hello World'	'HelloWorld'	1
'AA'	'AA'	0

You can also use the "Not equal" instruction to compare individual characters of a string (STRING). The number of the character to be compared is given in brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

When IEC Check is selected, the operands to be compared must have the same data type. If IEC Check is not selected, the width (length) of the operands must be the same. When floating-point numbers are compared, the operands to be compared must have the same data type regardless of the setting for the IEC Check.

Parameters

The following table shows the parameters of the "Not equal" instruction:

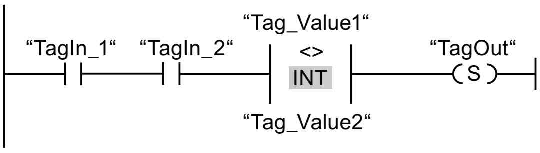
Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers, floating-point numbers, characters, TIME, DATE, TOD, DTL	I, Q, M, D, L or constant	First value to compare
<Operand2>	Input	Bit strings, integers, floating-point numbers, characters, TIME, DATE, TOD, DTL	I, Q, M, D, L or constant	Second value to compare

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have the signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" <> "Tag_Value2").

See also

Overview of the valid data types (Page 741)

CMP >=: Greater or equal

Description

You can use the "Greater or equal" instruction to determine if a first comparison value (<Operand1>) is greater than or equal to a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire current path as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) in the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) in the operand placeholder below the instruction.

The individual characters are compared by means of their ASCII code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character that is different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered greater.

The following table shows examples of string comparisons:

<Operand1>	<Operand2>	RLO of instruction
'BB'	'AA'	1
'AAA'	'AA'	1
'Hello World'	'Hello World'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0
'AAA'	'a'	0

You can also use the "Greater or equal" instruction to compare individual characters of a string (STRING). The number of the character to be compared is given in brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

When time values are compared, the RLO of the instruction is "1" if the point of time at <Operand1> is greater (more recent) than or equal to the point of time at <Operand2>.

Parameters

The following table shows the parameters of the "Greater or equal" instruction:

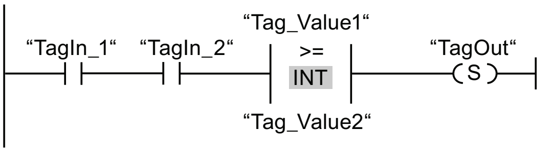
Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers, characters, TIME, DATE, TOD, DTL	I, Q, M, D, L or constant	First value to compare
<Operand2>	Input	Integers, floating-point numbers, characters, TIME, DATE, TOD, DTL	I, Q, M, D, L or constant	Second value to compare

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have the signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" >= "Tag_Value2").

See also

Overview of the valid data types (Page 741)

Example of detecting the fill level of a storage area (Page 1206)

CMP <=: Less or equal

Description

You can use the "Less or equal" instruction to determine if a first comparison value (<Operand1>) is less than or equal to a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire current path as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) in the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) in the operand placeholder below the instruction.

The individual characters are compared by means of their ASCII code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character that is different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string is considered smaller.

The following table shows examples of string comparisons:

<Operand1>	<Operand2>	RLO of instruction
'AA'	'aa'	1
'AAA'	'a'	1
'Hello World'	'Hello World'	1
'HelloWorld'	'Hello World'	0
'BB'	'AA'	0
'AAA'	'AA'	0

You can also use the "Less or equal" instruction to compare individual characters of a string (STRING). The number of the character to be compared is given in brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

When time values are compared, the RLO of the instruction is "1" if the point of time at <Operand1> is less (less recent) than or equal to the point of time at <Operand2>.

Parameters

The following table shows the parameters of the "Less or equal" instruction:

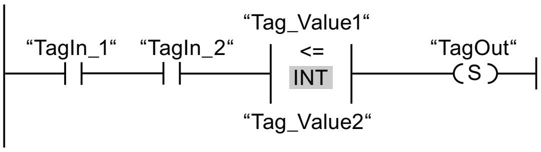
Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers, characters, TIME, DATE, TOD, DTL	I, Q, M, D, L or constant	First value to compare
<Operand2>	Input	Integers, floating-point numbers, characters, TIME, DATE, TOD, DTL	I, Q, M, D, L or constant	Second value to compare

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have the signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" <= "Tag_Value2").

See also

Overview of the valid data types (Page 741)

CMP >: Greater than

Description

You can use the "Greater than" instruction to determine if a first comparison value (<Operand1>) is greater than a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire current path as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) in the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) in the operand placeholder below the instruction.

The individual characters are compared by means of their ASCII code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character that is different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered greater.

The following table shows examples of string comparisons:

<Operand1>	<Operand2>	RLO of instruction
'BB'	'AA'	1
'AAA'	'AA'	1
'AA'	'aa'	0
'AAA'	'a'	0

You can also use the "Greater than" instruction to compare individual characters of a string (STRING). The number of the character to be compared is given in brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

When time values are compared, the RLO of the instruction is "1" if the point of time at <Operand1> is greater (more recent) than the point of time at <Operand2>.

Parameters

The following table shows the parameters of the "Greater than" instruction:

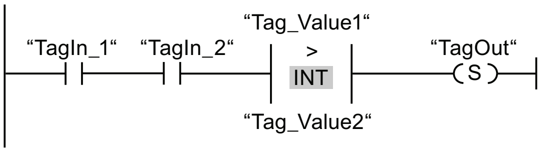
Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers, characters, TIME, DATE, TOD, DTL	I, Q, M, D, L or constant	First value to compare
<Operand2>	Input	Integers, floating-point numbers, characters, TIME, DATE, TOD, DTL	I, Q, M, D, L or constant	Second value to compare

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have the signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" > "Tag_Value2").

See also

Overview of the valid data types (Page 741)

CMP <: Less than

Description

You can use the "Less than" instruction to determine if a first comparison value (<Operand1>) is less than a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire current path as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) in the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) in the operand placeholder below the instruction.

The individual characters are compared by means of their ASCII code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character that is different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string is considered smaller.

The following table shows examples of string comparisons:

<Operand1>	<Operand2>	RLO of instruction
'AA'	'aa'	1
'AAA'	'a'	1
'BB'	'AA'	0
'AAA'	'AA'	0

You can also use the "Less than" instruction to compare individual characters of a string (STRING). The number of the character to be compared is given in brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

When time values are compared, the RLO of the instruction is "1" if the point of time at <Operand1> is less (less recent) than the point of time at <Operand2>.

Parameters

The following table shows the parameters of the "Less than" instruction:

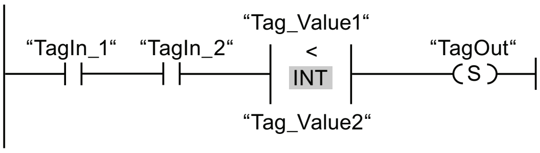
Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers, characters, TIME, DATE, TOD, DTL	I, Q, M, D, L or constant	First value to compare
<Operand2>	Input	Integers, floating-point numbers, characters, TIME, DATE, TOD, DTL	I, Q, M, D, L or constant	Second value to compare

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have the signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" < "Tag_Value2").

See also

Overview of the valid data types (Page 741)

Example of detecting the fill level of a storage area (Page 1206)

IN_RANGE: Value within range

Description

You can use the "Value within range" instruction to determine if the value at the VAL input is within a specific value range.

You specify the limits of the value range with the MIN and MAX inputs. The "Value within range" instruction compares the value at the VAL input with the values of the MIN and MAX inputs and sends the result to the box output. If the value at the VAL input fulfills the MIN <= VAL or VAL <= MAX comparison, the box output has the signal state "1". If the comparison is not fulfilled, the box output has the signal state "0".

If the box input has the signal state "0", the "Value within range" instruction is not executed.

The comparison function can only execute if the values to be compared are of the same data type and the box input is interconnected. You can also specify a constant for the MIN, MAX and VAL inputs.

Parameters

The following table shows the parameters of the "Value within range" instruction:

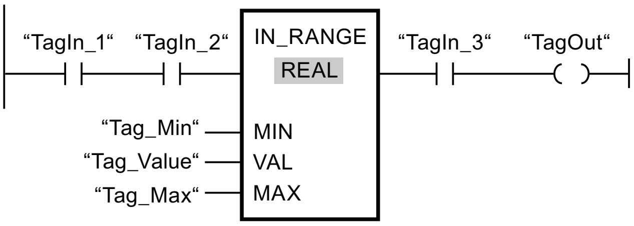
Parameters	Declaration	Data type	Memory area	Description
Box input	Input	BOOL	I, Q, M, D, L	Result of the previous logic operation
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VAL	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Comparison value
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
Box output	Output	BOOL	I, Q, M, D, L	Result of the comparison

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have the signal state "1".
- The value of operand ""Tag_Value"" is within the value range, which is specified by the current values of operands "Tag_Min" and "Tag_Max"(MIN <= VAL or VAL <= MAX).
- The operand "TagIn_3" has the signal state "1".

See also

Overview of the valid data types (Page 741)

OUT_RANGE: Value outside range

Description

You can use the "Value outside range" instruction to determine if the value at the VAL input is outside a specific value range.

You specify the limits of the value range with the MIN and MAX inputs. The "Value outside range" instruction compares the value at the VAL input with the values of the MIN and MAX inputs and sends the result to the box output. If the value at the VAL input fulfills the MIN > VAL or VAL > MAX comparison, the box output has the signal state "1". The box output also has the signal state "1" if a specified operand with the REAL data type shows an invalid value.

The box output returns the signal state "0", if the value at input VAL does not satisfy the MIN > VAL or VAL > MAX condition.

If the box input has the signal state "0", the "Value outside range" instruction is not executed.

The comparison function can only execute if the values to be compared are of the same data type and the box input is interconnected. You can also specify a constant for the MIN, MAX and VAL inputs.

Parameters

The following table shows the parameters of the "Value outside range" instruction:

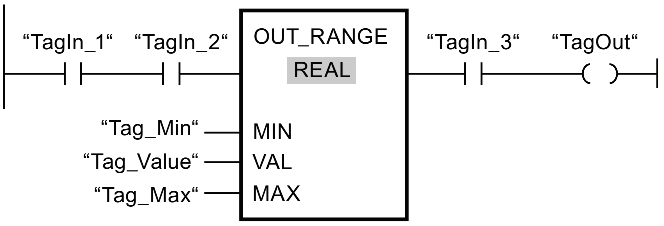
Parameters	Declaration	Data type	Memory area	Description
Box input	Input	BOOL	I, Q, M, D, L	Result of the previous logic operation
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VAL	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Comparison value
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
Box output	Output	BOOL	I, Q, M, D, L	Result of the comparison

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have the signal state "1".
- The value of operand "Tag_Value" is outside the value range, which is specified by the current values of operands "Tag_Min" and ""Tag_Max" (MIN > VAL or VAL > MAX).
- The operand "TagIn_3" has the signal state "1".

See also

Overview of the valid data types (Page 741)

----I OK I----: Check validity

Description

The "Check validity" instruction checks if the value of an operand (<Operand>) is a valid floating-point number. The query is started in each program cycle when the signal state at the input of the instruction is "1".

The output of the instruction has signal state "1" when the value of the operand is a valid floating-point number at the time of the query and the input of the instruction has signal state "1". In all other cases, the signal state at the output of the "Check validity" instruction is "0".

You can use the "Check validity" instruction together with the EN mechanism. If you connect the instruction box to an EN enable input, the enable input is set only when the result of the validity query of the value is positive. You can use this function to ensure that an instruction is enabled only when the value of the specified operand is a valid floating-point number.

Parameters

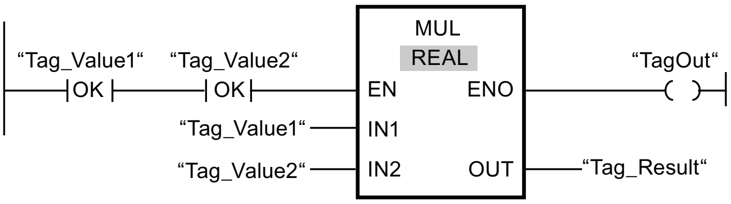
The following table shows the parameters of the "Check validity" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L or constant	Value to be queried.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the values of operands "Tag_Value1" and "Tag_Value2" are valid floating-point numbers, the "Multiply" (MUL) instruction is activated and the ENO output is set. When the "Multiply" (MUL) instruction is executed, the value of operand "Tag_Value1" is multiplied by the value of operand "Tag_Value2". The product of the multiplication is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO and "TagOut" outputs are set to signal state "1".

See also

Overview of the valid data types (Page 741)

----I NOT_OK I----: Check invalidity

Description

The "Check invalidity" instruction checks if the value of an operand (<Operand>) is an invalid floating-point number. The query is started in each program cycle when the signal state at the input of the instruction is "1".

The output of the instruction has signal state "1" when the value of the operand is an invalid floating-point number at the time of the query and the input of the instruction has signal state "1". In all other cases, the signal state at the output of the "Check invalidity" instruction is "0".

Parameters

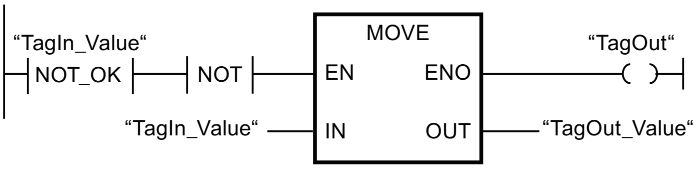
The following table shows the parameters of the "Check invalidity" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L or constant	Value to be queried.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the value at the operand "TagIn_Value" is an invalid floating-point number, the "Move value" (MOVE) instruction is not executed. The "TagOut" output is reset to signal state "0".

See also

Overview of the valid data types (Page 741)

Math functions

CALCULATE: Calculate

Description

You can use the "Calculate" instruction to define and execute an expression for calculating mathematical operations or complex logic combinations, depending on the selected data type.

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box. Depending on the data type selected, you can combine the functions of certain instructions to perform a complex calculation. The information for the expression to be calculated is entered in a dialog, which you can open with the icon at the upper right edge of the instruction box. The expression can contain names of input parameters and the syntax of the instructions. Operand names and operand addresses cannot be specified.

The following table shows the instructions that can be executed together in the expression of the "Calculate" instruction, depending on the selected data type:

Data type	Instruction	Syntax	Example
Bit strings	AND: AND logic operation	AND	IN1 AND IN2 OR IN3
	OR: OR logic operation	OR	
	XOR: EXCLUSIVE OR logic operation	XOR	
	INV: Create ones complement	NOT	
	SWAP: Swap	SWAP	
Integers	ADD: Add	+	(IN1 + IN2) * IN3; (ABS(IN2))*(ABS(IN1))
	SUB: Subtract	-	
	MUL: Multiply	*	
	DIV: Divide	/	
	MOD: Return remainder of division	MOD	
	INV: Create ones complement	NOT	
	NEG: Create twos complement	-(in1)	
	ABS: Form absolute value	ABS( )	
Floating-point numbers	ADD: Add	+	((SIN(IN2)*SIN(IN2)+(SIN(IN3)*SIN(IN3))/IN3; (SQR(SIN(IN2))+(SQR(COS(IN3))/IN2
	SUB: Subtract	-	
	MUL: Multiply	*	
	DIV: Divide	/	
	NEG: Create twos complement	**	
	ABS: Form absolute value	ABS( )	
	SQR : Form square	SQR( )	
	SQRT: Form square root	SQRT( )	
	LN : Form natural logarithm	LN( )	
	EXP : Form exponential value	EXP( )	
	FRAC: Return fraction	FRAC( )	
	SIN: Form sine value	SIN( )	

Data type	Instruction	Syntax	Example
	COS: Form cosine value	COS( )	
	TAN: Form tangent value	TAN( )	
	ASIN: Form arcsine value	ASIN( )	
	ACOS: Form arccosine value	ACOS( )	
	ATAN: Form arctangent value	ATAN( )	
	NEG: Create twos complement	-(in1)	
	TRUNC: Truncate numerical value	TRUNC( )	
	ROUND: Round numerical value	ROUND( )	
	CEIL: Generate next higher integer from floating-point number	CEIL( )	
	FLOOR: Generate next lower integer from floating-point number	FLOOR( )	

In its initial state, the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box.

The values of the input parameters are used to execute the specified expression. Not all of the defined input parameters have to be used in the expression. The result of the instruction is transferred to the output OUT.

If you use inputs in the expression that are not available in the box, they are inserted automatically. This requires that there are no gaps in the numbering of the inputs to be newly defined in the expression. For example, you cannot use the IN4 input in the expression unless the IN3 input has been defined.

The "Calculate" instruction is only executed if the signal state at the enable input EN is "1". When all the individual instructions of the specified expression are executed without errors, the ENO enable output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The result of the "Calculate" instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.
- An error occurred during execution of one of the instructions in the expression.

Parameters

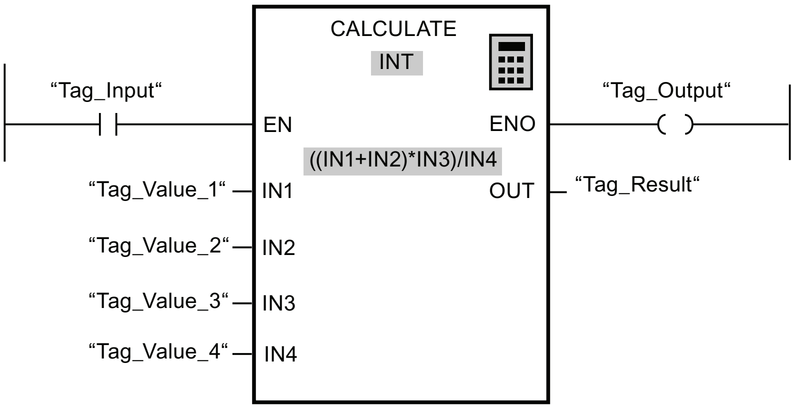
The following table shows the parameters of the "Calculate" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L or constant	First available input
IN2	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L or constant	Second available input
INn	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L or constant	Additionally inserted inputs
OUT	Output	Bit strings, integers, floating-point numbers	I, Q, M, D, L	Output to which the end result is to be transferred.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	"Tag_Value_1"	4
IN2	"Tag_Value_2"	4
IN3	"Tag_Value_3"	3

Parameters	Operand	Value
IN4	"Tag_Value_4"	2
OUT	"Tag_Result"	12

If the "Tag_Input" input has the signal state"1", the "Calculate" instruction is executed. The value of operand "Tag_Value_1" is added to the value of operand "Tag_Value_2". The sum is multiplied with the value of operand "Tag_Value_3". The product is divided by the value of operand "Tag_Value_4". The quotient is transferred as end result to the operand "Tag_Result" at the OUT output of the instruction. If the instruction is executed without errors, the ENO output and operand "Tag_Output" are set to "1".

See also

Using the "Calculate" instruction (Page 920)

Overview of the valid data types (Page 741)

Inserting additional inputs and outputs in LAD elements (Page 927)

Removing inputs and outputs (Page 928)

Basics of the EN/ENO mechanism (Page 819)

ADD: Add

Description

You can use the "Add" instruction to add the value at input IN1 and the value at input IN2 and query the sum at output OUT (OUT = IN1+IN2).

In its initial state, the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are added. The sum is stored at the OUT output.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO enable output also returns the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Add" instruction:

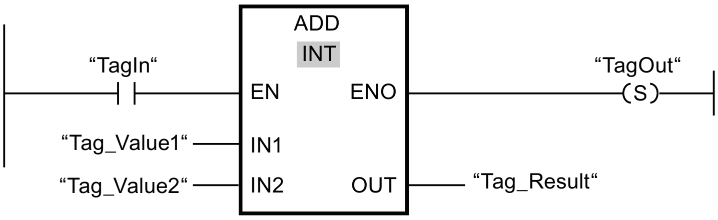
Parameters		Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	First number to be added
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Second number to be added
INn	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Optional input values that are added.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Sum

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Add" instruction is executed. The value of operand "Tag_Value1" is added to the value of operand "Tag_Value2". The result of the addition is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Removing inputs and outputs (Page 928)

Basics of the EN/ENO mechanism (Page 819)

Inserting additional inputs and outputs in LAD elements (Page 927)

SUB: Subtract

Description

You can use the "Subtract" instruction to subtract the value at the IN2 input from the value at the IN1 input and query the difference at the OUT output (OUT = IN1-IN2).

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Subtract" instruction:

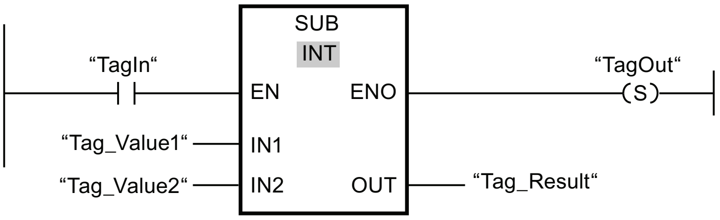
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Minuend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Subtrahend
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Difference

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Subtract" instruction is executed. The value of operand "Tag_Value2" is subtracted from the value of operand "Tag_Value1". The result of the subtraction is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

MUL: Multiply

Description

You can use the "Multiply" instruction to multiply the value at the IN1 input by the value at the IN2 input and query the product at the OUT output (OUT = IN1*IN2).

The number of inputs can be expanded in the instruction box. The added inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are multiplied. The product is stored at the OUT output.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The result is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Multiply" instruction:

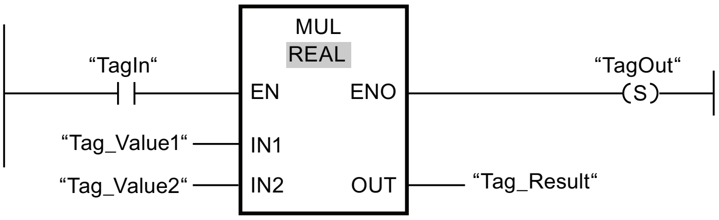
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Multiplier
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Multiplicand
INn	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Optional input values that can be multiplied.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Product

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Multiply" instruction is executed. The value of operand "Tag_Value1" is multiplied by the value of operand "Tag_Value2". The result of the multiplication is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Removing inputs and outputs (Page 928)

Basics of the EN/ENO mechanism (Page 819)

Inserting additional inputs and outputs in LAD elements (Page 927)

DIV: Divide

Description

You can use the "Divide" instruction to divide the value at the IN1 input by the value at the IN2 input and query the quotient at the OUT output (OUT = IN1/IN2).

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Divide" instruction:

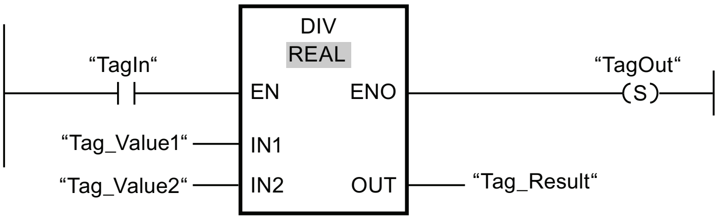
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Dividend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Divisor
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Quotient value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Divide" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The division result is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

MOD: Return remainder of division

Description

You can use the "Return remainder of division" instruction to divide the value at the IN1 input by the value at the IN2 input and query the remainder at the OUT output.

The instruction is only executed if the signal state is "1" at the EN input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The instruction is not executed if the signal state at the EN input is "0". In this case, the ENO output is reset.

Parameters

The following table shows the parameters of the "Return remainder of division" instruction:

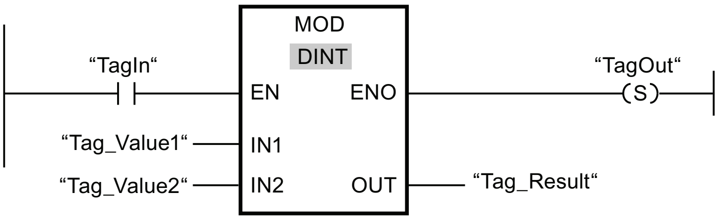
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers	I, Q, M, D, L or constant	Dividend
IN2	Input	Integers	I, Q, M, D, L or constant	Divisor
OUT	Output	Integers	I, Q, M, D, L	Remainder of division

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Return remainder of division" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The remainder is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

NEG: Create twos complement

Description

You can use the "Create twos complement" instruction to change the sign of the value at the IN input and query the result at the OUT output. If there is a positive value at the IN input, for example, the negative equivalent of this value is sent to the OUT output.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Create twos complement" instruction:

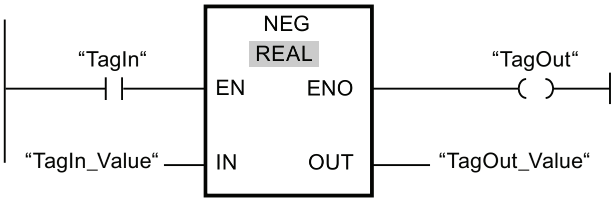
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	SINT, INT, DINT, floating-point numbers	I, Q, M, D, L or constant	Input value
OUT	Output	SINT, INT, DINT, floating-point numbers	I, Q, M, D, L	Twos complement of the input value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Create twos complement" instruction is executed. The sign of the value at the "TagIn_Value" input is changed and the result is provided at the "TagOut_Value" output. If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

INC: Increment

Description

You can use the "Increment" instruction to change the value of the operand at the IN/OUT parameter to the next higher value and query the result. The "Increment" instruction is only started when the signal state at the EN enable input is "1". If no overflow error occurs during the execution, the ENO output also has the signal state "1".

If the signal state is "0" at the EN enable input, the instruction is not executed. In this case, the ENO enable output is reset.

Parameters

The following table shows the parameters of the "Increment" instruction:

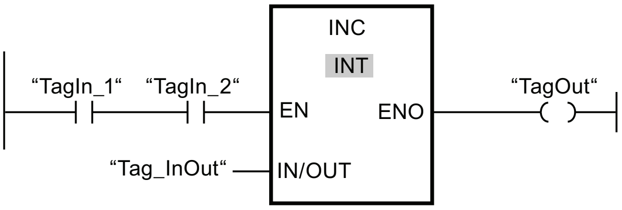
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN/OUT	InOut	Integers	I, Q, M, D, L	Value to be incremented.

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operands "TagIn_1" and "TagIn_2" have the signal state "1", the value of operand "Tag_InOut" is incremented by one and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

DEC: Decrement

Description

You can use the "Decrement" instruction to change the value of the operand at the IN/OUT parameter to the next lower value and query the result. Execution of the "Decrement" instruction is started when the signal state at the EN enable input is "1". If the range of values of the selected data type is not exceeded during processing, the ENO output also has the signal state "1".

If the signal state is "0" at the EN enable input, the instruction is not executed. In this case, the ENO enable output is reset.

Parameters

The following table shows the parameters of the "Decrement" instruction:

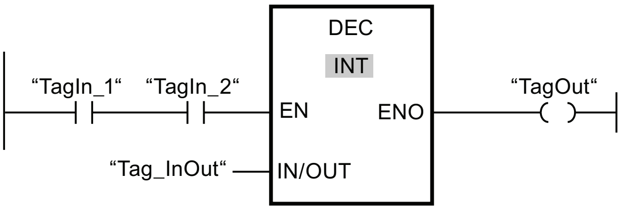
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN/OUT	InOut	Integers	I, Q, M, D, L	Value to be decremented.

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operands "TagIn_1" and "TagIn_2" have the signal state "1", the value of operand "Tag_InOut" is decremented by one and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

ABS: Form absolute value

Description

The "Form absolute value" instruction calculates the absolute value of the value specified at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Form absolute value" instruction:

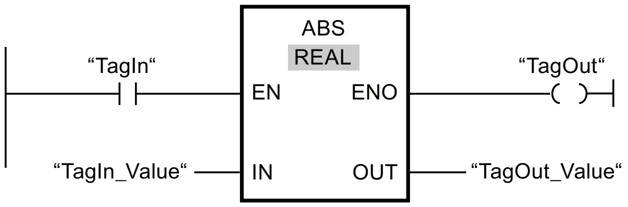
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	SINT, INT, DINT, floating-point numbers	I, Q, M, D, L or constant	Input value
OUT	Output	SINT, INT, DINT, floating-point numbers	I, Q, M, D, L	Absolute value of the input value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	"TagIn_Value"	- 6, 234
OUT	"TagOut_Value"	6, 234

If operand "TagIn" has the signal state "1", the "Form absolute value" instruction is executed. The instruction calculates the absolute value of the value at the "TagIn_Value" input and sends the result to the "TagOut_Value" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

MIN: Get minimum

Description

The "Get minimum" instruction compares the values of the available inputs and writes the lowest value to the OUT output. The number of inputs can be expanded in the instruction box by additional inputs. The inputs are numbered in ascending order in the box.

A minimum of two and a maximum of 100 inputs must be specified for the execution of the instruction.

Execution of the instruction presupposes that the tags at all inputs are of the same type and that enable input EN has signal state "1". If the instruction is executed without errors, the ENO enable output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get minimum" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Second input value

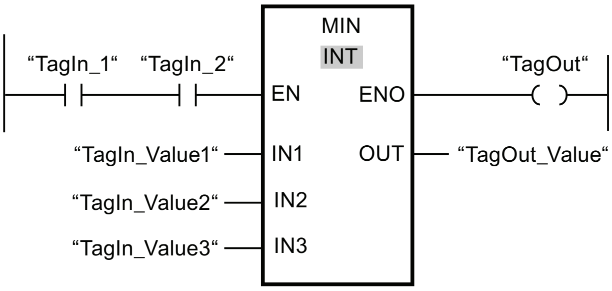
Parameters	Declaration	Data type	Memory area	Description
INn	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Additionally inserted inputs whose values are to be compared.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	TagIn_Value1	12 222
IN2	TagIn_Value2	14 444
IN3	TagIn_Value3	13 333
OUT	TagOut_Value	12 222

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Get minimum" instruction is executed. The instruction compares the values of the specified operands and copies the lowest value ("TagIn_Value1") to the "TagOut_Value" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Inserting additional inputs and outputs in LAD elements (Page 927)

Basics of the EN/ENO mechanism (Page 819)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values of the available inputs and writes the highest value to the OUT output. The number of inputs can be expanded in the instruction box by additional inputs. The inputs are numbered in ascending order in the box.

A minimum of two and a maximum of 100 inputs must be specified for the execution of the instruction.

Execution of the instruction presupposes that the tags at all inputs are of the same type and that enable input EN has signal state "1". If the instruction is executed without errors, the ENO enable output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get maximum" instruction:

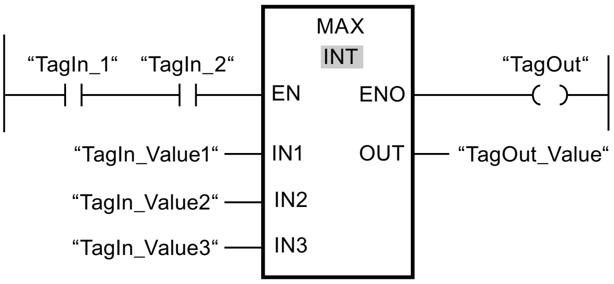
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Second input value
INn	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Additionally inserted inputs whose values are to be compared.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	TagIn_Value1	12 222
IN2	TagIn_Value2	14 444
IN3	TagIn_Value3	13 333
OUT	TagOut_Value	14 444

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Get maximum" instruction is executed. The instruction compares the values of the specified operands and copies the highest value ("TagIn_Value2") to the "TagOut_Value" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Inserting additional inputs and outputs in LAD elements (Page 927)

Removing inputs and outputs (Page 928)

Basics of the EN/ENO mechanism (Page 819)

LIMIT: Set limit value

Description

The "Set limit value" instruction limits the value at the IN input to the values at the MN and MX inputs. If the value at the IN input meets the MN < IN < MX condition, it is copied to the OUT output. If the condition is not fulfilled and the input value IN is below the low limit MN, the output OUT is set to the value of the MN input. If the high limit is exceeded, the MX output OUT is set to the value of the MX input.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO enable output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value at the MN input is greater than the value at the MX input.

Parameters

The following table shows the parameters of the "Set limit value" instruction:

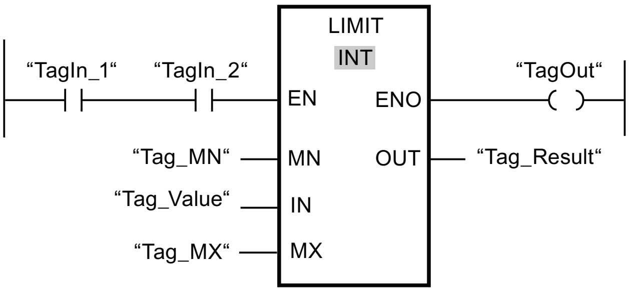
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit
IN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Input value
MX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
MN	Tag_MN	12 000
IN	Tag_Value	8 000
MX	Tag_MX	16 000
OUT	Tag_Result	12 000

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Set limit value" instruction is executed. The value of operand "Tag_Value" is compared with the values of operands "Tag_MN" and "Tag_MX". Since the value of operand "Tag_Value" is less than the low limit, the value of operand "Tag_MN" is copied to the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

SQR: Form square

Description

You can use the "Form square" instruction to square the value at the IN input and query the result at the OUT output.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form square" instruction:

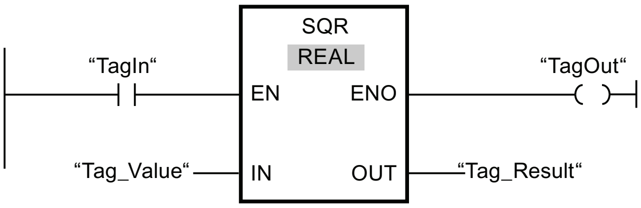
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L	Square of the input value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	5.0
OUT	Tag_Result	25.0

If operand "TagIn" has the signal state "1", the "Form square" instruction is executed. The instruction squares the value of operand "Tag_Value" and sends the result to the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

SQRT: Form square root

Description

You can use the "Form square root" instruction to find the square root of the value at the IN input and query the result at the OUT output. The instruction outputs a positive result if the input value is greater than zero. If input values are less than zero, the OUT output returns an invalid floating-point number. If the value at input IN is "0", then the result is also "0".

The instruction is only executed if the signal state is "1" at the enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is negative.

Parameters

The following table shows the parameters of the "Form square root" instruction:

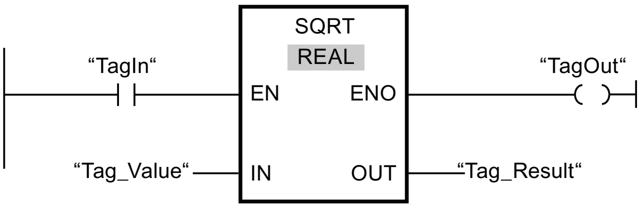
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L	Square root of the input value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	25.0
OUT	Tag_Result	5.0

If operand "TagIn" has the signal state "1", the "Form square root" instruction is executed. The instruction finds the square root of the value of operand "Tag_Value" and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

LN: Form natural logarithm

Description

The "Form natural logarithm" instruction calculates the natural logarithm to base e (e = 2.718282e+00) of the value at input IN. The result is sent to the OUT output and can be queried there. The instruction outputs a positive result if the input value is greater than zero. If input values are less than zero, the OUT output returns an invalid floating-point number.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is processed without errors, the ENO enable output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is negative.

Parameters

The following table shows the parameters of the "Form natural logarithm" instruction:

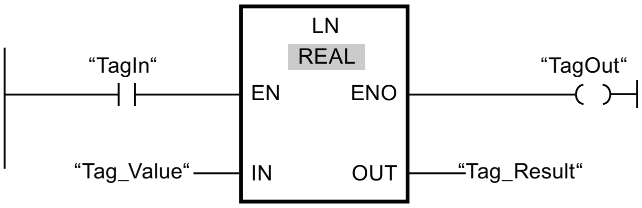
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L	Natural logarithm of the input value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Form natural logarithm" instruction is executed. The instruction forms the natural logarithm of the value at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

EXP: Form exponential value

Description

The "Form exponential value" instruction calculates the exponent from the base e (e = 2.718282e+00) and the value specified at the IN input. The result is sent to the OUT output and can be queried there (OUT = e^{IN}).

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form exponential value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

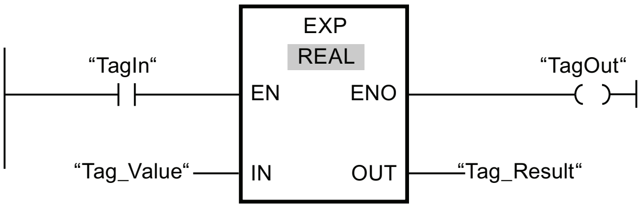
Parameters	Declaration	Data type	Memory area	Description
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L	Exponential value of input value IN

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state"1", the "Form exponential value" instruction is executed. The instruction calculates the exponent from base e and the value of operand "Tag_Value" and sends the result to the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

SIN: Form sine value

Description

Use the "Form sine value" instruction to calculate the sine of the angle. The size of the angle is specified in radians at the input IN. The result of the instruction is sent to the OUT output and can be queried there.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form sine value" instruction:

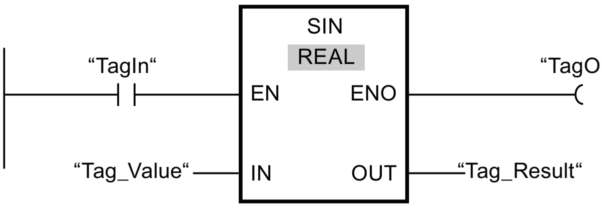
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L	Sine of the specified angle

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796e+00 (π/2)
OUT	Tag_Result	1.0

If operand "TagIn" has the signal state "1", the "Form sine value" instruction is executed. The instruction calculates the sine of the angle specified at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

COS: Form cosine value

Description

You can use the "Form cosine value" instruction to calculate the cosine of an angle. The size of the angle is specified in radians at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form cosine value" instruction:

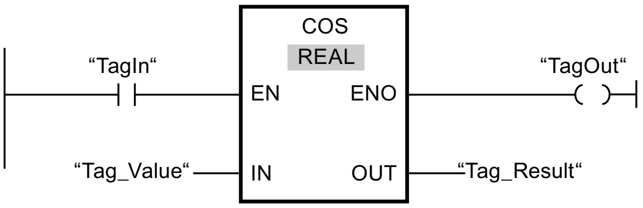
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L	Cosine of the specified angle

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796e+00 (π/2)
OUT	Tag_Result	0

If operand "TagIn" has the signal state "1", the "Form cosine value" instruction is executed. The instruction calculates the cosine of the angle specified at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

TAN: Form tangent value

Description

The "Form tangent value" instruction calculates the tangent of an angle. The size of the angle is specified in radians at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form tangent value" instruction:

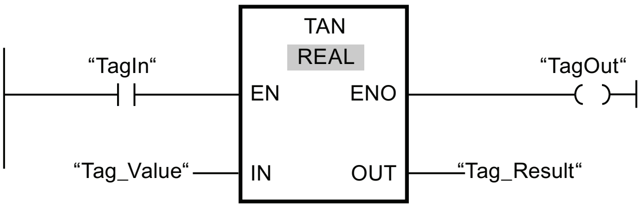
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L	Tangent of the specified angle

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+3.141593e+00 (π)
OUT	Tag_Result	0

If operand "TagIn" has the signal state "1", the "Form tangent value" instruction is executed. The instruction calculates the tangent of the angle specified at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

ASIN: Form arcsine value

Description

The "Form arcsine value" instruction calculates the size of the angle corresponding to the sine value specified at the IN input. Only valid floating-point numbers within the range -1 to +1 can be specified at the IN input. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is outside the permitted value range (-1 to +1).

Parameters

The following table shows the parameters of the "Form arcsine value" instruction:

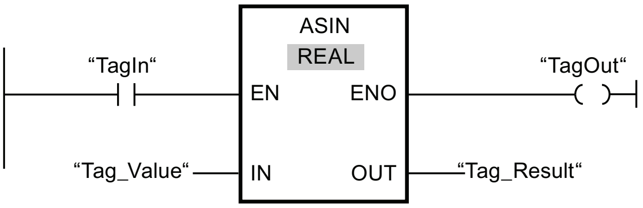
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Sine value
OUT	Output	Floating-point numbers	I, Q, M, D, L	Size of angle in radians

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+1.570796e+00 (π/2)

If operand "TagIn" has the signal state "1", the "Form arcsine value" instruction is executed. The instruction calculates the size of the angle corresponding to the sine value at the "Tag_Value" input. The result of the instruction is stored in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

ACOS: Form arccosine value

Description

The "Form arccosine value" instruction calculates the size of the angle corresponding to the cosine value specified at the IN input. Only valid floating-point numbers within the range -1 to +1 can be specified at the IN input. The calculated angle size is output in radians at the OUT output and can range in value from 0 to +π.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is outside the permitted value range (-1 to +1).

Parameters

The following table shows the parameters of the "Form arccosine value" instruction:

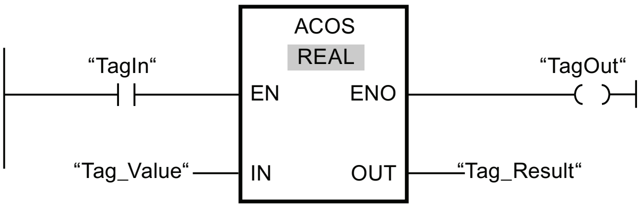
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Cosine value
OUT	Output	Floating-point numbers	I, Q, M, D, L	Size of angle in radians

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	0
OUT	Tag_Result	+1.570796e+00 (π/2)

If operand "TagIn" has the signal state "1", the "Form arccosine value" instruction is executed. The instruction calculates the size of the angle corresponding to the cosine value at the "Tag_Value" input. The result of the instruction is stored in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

ATAN: Form arctangent value

Description

The "Form arctangent value" instruction calculates the size of the angle corresponding to the tangent value specified at the IN input. Only valid floating-point numbers may be specified at the IN input. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form arctangent value" instruction:

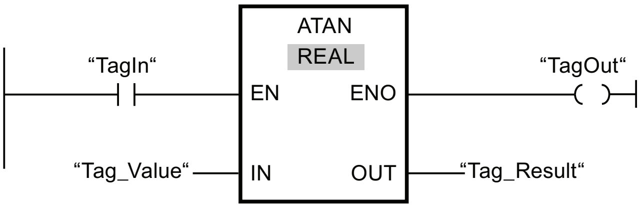
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Tangent value
OUT	Output	Floating-point numbers	I, Q, M, D, L	Size of angle in radians

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+0.785398e+00 (π/4)

If operand "TagIn" has the signal state"1", the "Form arctangent value" instruction is executed. The instruction calculates the size of the angle corresponding to the tangent value at the "Tag_Value" input. The result of the instruction is stored in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

FRAC: Return fraction

Description

You can use the "Return fraction" instruction to determine the decimal places of the value at the IN input. The result of the query is stored at the OUT output and can be queried there. If the value at the IN input is 123.4567, for example, the OUT output returns the value 0.4567. The instruction is started when the EN input has the signal state "1". In this case, the ENO enable output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- Errors occur during execution of the instruction, for example, there is no valid floating-point number at the IN input.

Parameters

The following table shows the parameters of the "Return fraction" instruction:

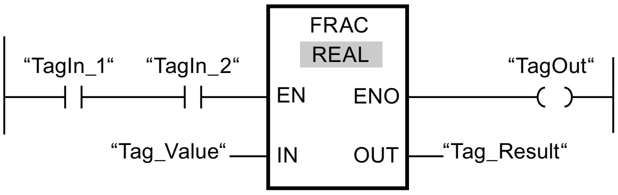
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Value, whose decimal places are to be determined.
OUT	Output	Floating-point numbers	I, Q, M, D, L	Decimal places of the value at the IN input

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	2,555
OUT	Tag_Result	0,555

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Return fraction" instruction is started. The decimal places from the value of operand "Tag_Value" are copied to operand "Tag_Result". If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

EXPT: Exponentiate

Description

You can use the "Exponentiate" instruction to raise the value at the IN1 input to a power specified with the value at the IN2 input. The result of the instruction is stored in the OUT output and can be queried there (OUT = $IN1^{IN2}$).

The IN1 input can only be assigned valid floating-point numbers. Integers can also be assigned to the IN2 input.

The "Exponentiate" instruction can only be executed when the signal state at the EN enable input is "1". In this case, the ENO enable output has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- Errors occur during execution of the instruction, for example, an overflow occurs.

Parameters

The following table shows the parameters of the "Exponentiate" instruction:

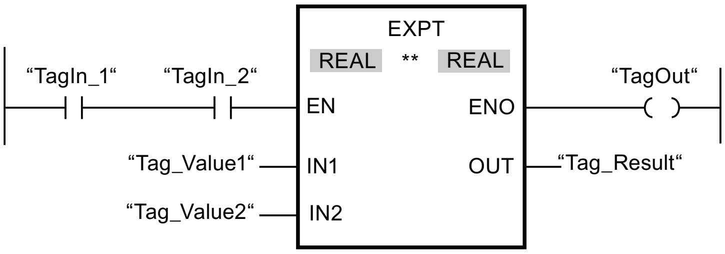
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Floating-point numbers	I, Q, M, D, L or constant	Base value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Value with which the base value is exponentiated
OUT	Output	Floating-point numbers	I, Q, M, D, L	Result

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Exponentiate" instruction is started. The value of operand "Tag_Value1" is raised to a power specified with the value of operand "Tag_Value2". The result is stored in the "Tag_Result" output. If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

Move operations

MOVE: Move value

Description

You use the "Move value" instruction to transfer the contents of the operand at the IN input to the operand at the OUT1 output. The transfer is always made in the direction of the ascending address.

The following table shows the possible transfers:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE ,TOD, CHAR
WORD	WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD, CHAR
DWORD	DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, CHAR
SINT	SINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
USINT	USINT, UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
INT	INT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UINT	UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
DINT	DINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UDINT	UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
REAL	REAL	DWORD, REAL
LREAL	LREAL	LREAL
TIME	TIME	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME
DATE	DATE	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, DATE
TOD	TOD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TOD
DTL	DTL	DTL
CHAR	CHAR	BYTE, WORD, DWORD, characters of a string1)
Characters of a string1)	Characters of a string	CHAR, characters of a string
ARRAY2)	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT

1) You can also use the "Move value" instruction to transfer individual characters of a string (STRING) to operands of data type CHAR. The number of the character to be transferred is given in brackets next to the operand name. "MyString[2]", for example, transfers the second character of the "MyString" string. You can also transfer an operand with the CHAR data type to individual characters of a string. You can also replace a specific character of a string with the character of another string.

2) The transfer of whole arrays is only possible if the data type of the array components of the operand at the IN input matches that of the operand at the OUT1 output.

If the bit length of the data type at the IN input exceeds the bit length of the data type at the OUT1 output, the more significant bits of the source value are lost. If the bit length of the data type at the IN input is less than the bit length of the data type at the OUT1 output, then the more significant bits of the destination value will be overwritten with zeros.

In its initial state, the instruction box contains 1 output (OUT1). The number of outputs can be extended. The added outputs are numbered in ascending order on the box. When the instruction is executed, the content of the operand at the IN input is sent to all available outputs. The instruction box cannot be extended if structured data types (DTL, STRUCT, ARRAY) or characters of a string (STRING) are transferred.

The instruction is only executed if the signal state is "1" at the EN enable input. In this case, the ENO output also has the signal state "1".

If the signal state at the EN input is "0", the ENO enable output is reset to "0".

The "Move block" (MOVE_BLK) and "Move block uninterruptible" (UMOVE_BLK) instructions can also be used to copy operands of the ARRAY data type. Operands of the STRING data type can be copied using the S_MOVE instruction.

Parameters

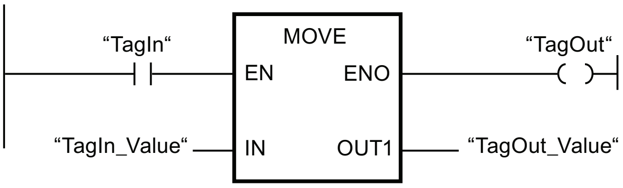
The following table shows the parameters of the "Move value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers, floating-point numbers, timers, DATE, TIME, TOD, DTL, CHAR, STRUCT, ARRAY	I, Q, M, D, L or constant	Source value
OUT1	Output	Bit strings, integers, floating-point numbers, timers, DATE, TIME, TOD, DTL, CHAR, STRUCT, ARRAY	I, Q, M, D, L	Operands to which in the source value is transferred.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
OUT1	TagOut_Value	0011 1111 1010 1111

If operand "TagIn" has the signal state"1", the "Move value" instruction is executed. The instruction copies the contents of operand "TagIn_Value" to operand "TagOut_Value" and sets output "TagOut" to signal state "1".

See also

Overview of the valid data types (Page 741)

Removing inputs and outputs (Page 928)

Basics of the EN/ENO mechanism (Page 819)

MOVE_BLK: Move block (Page 1330)

UMOVE_BLK: Move block uninterruptible (Page 1332)

S_MOVE: Move character string (Page 1678)

Inserting additional inputs and outputs in LAD elements (Page 927)

FieldRead: Read field

Description

You can use the "Read field" instruction to read out a specific component from the field specified in the MEMBER parameter and transfer its content to the tag in the VALUE parameter. You specify the index of the field component to be read in the INDEX parameter. Specify the first component of the field from which reading is to occur in the MEMBER parameter.

The data type of the field component in the MEMBER parameter and the data type of the tags in the VALUE parameter must match the data type of the "Read field" instruction.

The execution of the "Read field" instruction is only started when the signal state at the EN enable input is "1". If no errors occur during execution, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The field component specified in the INDEX parameter is not defined in the field specified in the MEMBER parameter.
- Errors such as an overflow occur during execution.

Parameters

The following table shows the parameters of the "Read field" instruction:

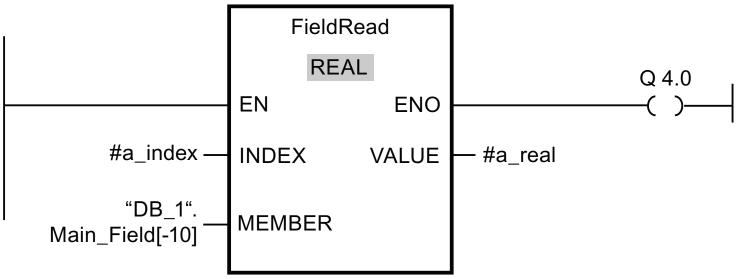
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
INDEX	Input	DINT	I, Q, M, D, L or constant	Index of field component whose content is read out
MEMBER	Input	BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR as a component of an ARRAY tag	I, Q, M, D, L	First component of the field from which reading occurs
VALUE	Output	BOOL. bit strings, integers, floating-point numbers, timers, DATE and CHAR	I, Q, M, D, L	Operand to which the content of the field component is transferred

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Tag	Value
INDEX	a_index	4
MEMBER	"DB_1".Main_Field[-10]	First component of the "Main_Field[-10..10] of REAL" field in the "DB_1" data block
VALUE	a_real	Component with index 4 of the "Main_Field[-10..10] of REAL" field

The field component with index 4 is read from the "Main_Field[-10...10] of REAL" field and written to the "a_real" tag. The field component to be read is specified by the value at the INDEX parameter.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

FieldWrite: Write field

Description

You can use the "Write field" instruction to transfer the content of the tag in the VALUE parameter to a specific component of the field in the MEMBER parameter. You specify the index of the field component that is being written to by the value in the INDEX parameter. At the MEMBER parameter, specify the first component of the field which is to be written to.

The data type of the field component specified in the MEMBER parameter and the data type of the tag in the VALUE parameter must match the data type of the "Write field" instruction.

The execution of the "Write field" instruction is only started when the signal state at the EN enable input is "1". If no errors occur during execution, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The field component specified in the INDEX parameter is not defined in the field specified in the MEMBER parameter.
- Errors such as an overflow occur during execution.

Parameters

The following table shows the parameters of the "Write field" instruction:

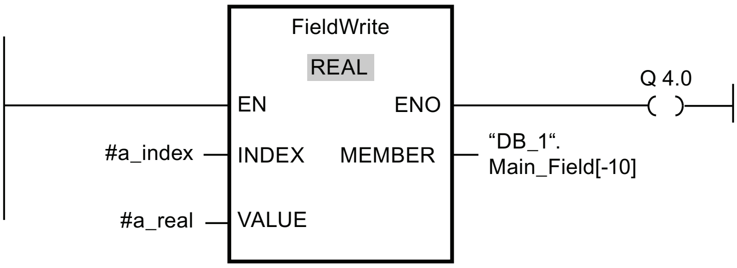
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
INDEX	Input	DINT	I, Q, M, D, L or constant	Index of the field component that is being written with the content of VALUE
VALUE	Input	BOOL. bit strings, integers, floating-point numbers, timers, DATE and CHAR	I, Q, M, D, L or constant	Operand whose content is copied
MEMBER	Output	BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR as a component of an ARRAY tag	I, Q, M, D, L	First component of the field to which the content of VALUE is written

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
INDEX	a_index	4
VALUE	a_real	10,54
MEMBER	"DB_1".Main_Field[-10]	First component of the "Main_Field[-10..10] of REAL" field in the "DB_1" data block

The value "10,54" of the "a_real" tag is written to the field component with index 4 of the "Main_Field[-10..10] of REAL" field. The index of the field component to which the content of the "a_real" tag is transferred" is specified by the value in the INDEX parameter.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

MOVE_BLK: Move block

Description

You can use the "Move block" instruction to move the contents of a memory area (source area) to another memory area (destination area). The number of elements to be copied to the destination area is specified with the COUNT parameter. The width of the elements to be copied is defined by the width of the element at the IN input. The copy operation takes place in the direction of ascending addresses.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- More data is copied than is made available at the IN input or OUT output.

If the last BOOL element of an ARRAY structure is not at a byte limit (for example, bit 16 of 2 bytes) and an overflow occurs, the ENO output remains at "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO output is reset to "0".

Parameters

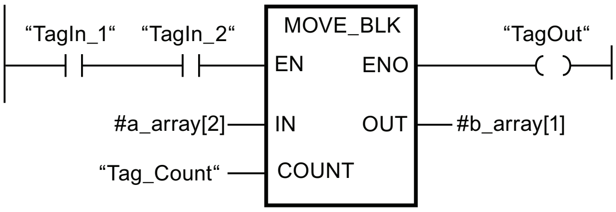
The following table shows the parameters of the "Move block" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Binary numbers, integers, floating-point numbers, timers, DATE and CHAR as a component of an ARRAY structure	D, L	The first element of the source area that is being copied from.
COUNT	Input	UINT	I, Q, M, D, L or constant	Number of elements to be copied from the source area to the destination area.
OUT	Output	Binary numbers, integers, floating-point numbers, timers, DATE and CHAR as a component of an ARRAY structure	D, L	The first element of the destination area to which the contents of the source area are being copied.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	a_array[2]	Operand "a_array" has ARRAY data type and consists of 5 elements of the INT data type.
COUNT	Tag_Count	3
OUT	b_array[1]	Operand "b_array" has ARRAY data type and consists of 6 elements of the INT data type.

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Move block" instruction is executed. The instruction selects three INT elements from the "a_array" (a_array[2..4]) tag and copies their contents into the "b_array" (b_array[1..3]) output tag. If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

Inserting additional inputs and outputs in LAD elements (Page 927)

UMOVE_BLK: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to copy the contents of a memory area (source area) to another memory area (destination area) without interruption. The number of elements to be copied to the destination area is specified with the COUNT parameter. The width of the elements to be copied is defined by the width of the element at the IN input. The content of the source area is copied to the destination area in the direction of the ascending address.

Note

The copy operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the "Move block uninterruptible" instruction.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- More data is copied than is made available at the IN input or OUT output.

If the last BOOL element of an ARRAY structure is not at a byte limit (for example, bit 16 of 2 bytes) and an overflow occurs, the ENO output remains at "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO output is reset to "0".

Parameters

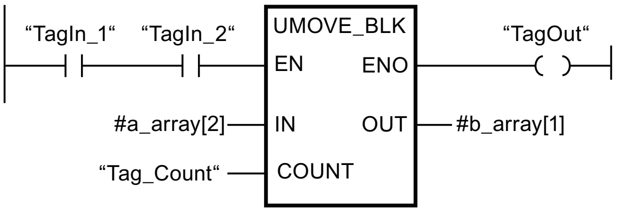
The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Binary numbers, integers, floating-point numbers, timers, DATE and CHAR as a component of an ARRAY structure	D, L	The first element of the source area that is being copied from.
COUNT	Input	UINT	I, Q, M, D, L or constant	Number of elements to be copied from the source area to the destination area.
OUT	Output	Binary numbers, integers, floating-point numbers, timers, DATE and CHAR as a component of an ARRAY structure	D, L	The first element of the destination area to which the contents of the source area are being copied.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	a_array[2]	Operand "a_array" has ARRAY data type and consists of 5 elements of the INT data type.
COUNT	Tag_Count	3
OUT	b_array[1]	The b_array tag has ARRAY data type and consists of 6 elements of the INT data type.

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Move block uninterruptible" instruction is executed. The instruction selects three INT elements from the "a_array" (a_array[2..4]) tag and copies their contents into the "b_array" (b_array[1..3]) output tag. The copy operation cannot be interrupted by other operating system activities. If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

Inserting additional inputs and outputs in LAD elements (Page 927)

FILL_BLK: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the value of the IN input. The destination area is filled starting from the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the IN input is selected and copied to the destination area as often as specified by the value in the COUNT parameter.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- More data is copied than is made available at the IN input or OUT output.

If the last BOOL element of an ARRAY structure is not at a byte limit (for example, bit 16 of 2 bytes) and an overflow occurs, the ENO output remains at "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO output is reset to "0".

Parameters

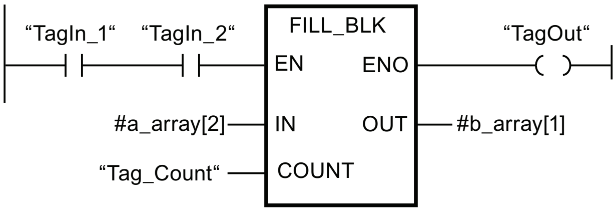
The following table shows the parameters of the "Fill block" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Binary numbers, integers, floating-point numbers, timers, DATE and CHAR as a component of an ARRAY structure	D, L or constant	Element used to fill the destination area.
COUNT	Input	UINT	I, Q, M, D, L or constant	Number of repeated copy operations
OUT	Output	Binary numbers, integers, floating-point numbers, timers, DATE and CHAR as a component of an ARRAY structure	D, L	Address in destination area from which filling starts.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	A_array[2]	Operand "a_array" has ARRAY data type and consists of 4 elements of the WORD data type (ARRAY[1..4] of WORD).
COUNT	Tag_Count	3
OUT	b_array[1]	Operand "b_array" has ARRAY data type and consists of 5 elements of the WORD data type (ARRAY[1..5] of WORD).

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Fill block" instruction is executed. The instruction copies the second element (a_array[2]) of the "a_array" tag three times to the "b_array" output tag (b_array[1..3]). If the instruction is executed without errors, the ENO and "TagOut" outputs are set to signal state "1".

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

Inserting additional inputs and outputs in LAD elements (Page 927)

UFILL_BLK: Fill block uninterruptible

Description

You can use the "Fill block uninterruptible" instruction to fill a memory area (destination area) with the value of the IN input without interruption. The destination area is filled starting from the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the IN input is selected and copied to the destination area as often as specified by the value in the COUNT parameter.

Note

The copy operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the "Fill block uninterruptible" instruction.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- More data is copied than is made available at the IN input or OUT output.

If the last BOOL element of an ARRAY structure is not at a byte limit (for example, bit 16 of 2 bytes) and an overflow occurs, the ENO output remains at "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO output is reset to "0".

Parameters

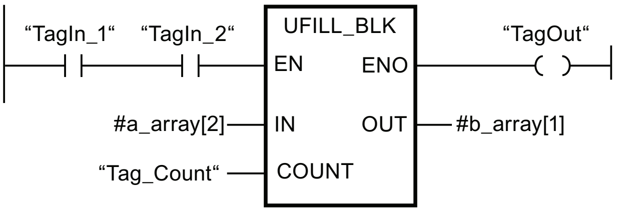
The following table shows the parameters of the "Fill block uninterruptible" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Binary numbers, integers, floating-point numbers, timers, DATE and CHAR as a component of an ARRAY structure	D, L or constant	Element used to fill the destination area.
COUNT	Input	UINT	I, Q, M, D, L or constant	Number of repeated copy operations
OUT	Output	Binary numbers, integers, floating-point numbers, timers, DATE and CHAR as a component of an ARRAY structure	D, L	Address in destination area from which filling starts.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	a_array[2]	Operand "a_array" has ARRAY data type and consists of 4 elements of the WORD data type (ARRAY[1..4] of WORD).
COUNT	Tag_Count	3
OUT	b_array[1]	Operand "b_array" has ARRAY data type and consists of 5 elements of the WORD data type (ARRAY[1..5] of WORD).

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Fill block uninterruptible" instruction is executed. The instruction copies the second element (a_array[2]) of the "a_array" tag three times to "b_array" (b_array[1..3]) output tag. The copy operation cannot be interrupted by other operating system activities. If the instruction is executed without errors, the ENO and "TagOut" outputs are set to signal state "1".

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

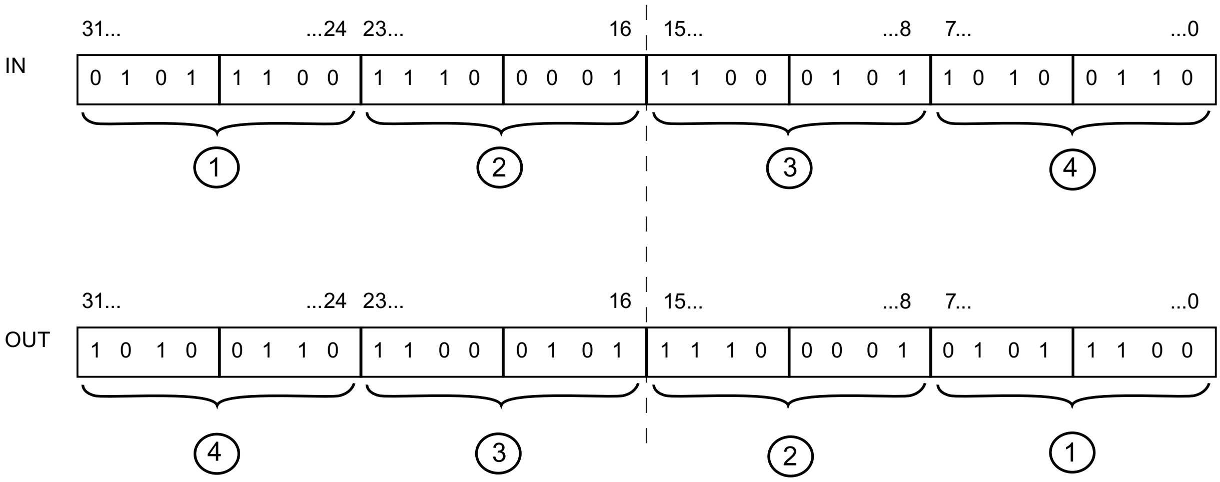
Inserting additional inputs and outputs in LAD elements (Page 927)

SWAP: Swap

Description

The "Swap" instruction changes the order of the bytes at the IN input and queries the result at the OUT output.

The following figure shows how the bytes of an operand of the DWORD data type are swapped using the "Swap" instruction:



The "Swap" instruction can only be executed when the signal state at the EN enable input is "1". In this case, the ENO enable output has the signal state "1".

The ENO enable output is reset when the EN enable input has the signal state "0" or errors occur during execution of the instruction.

Parameters

The following table shows the parameters of the "Swap" instruction:

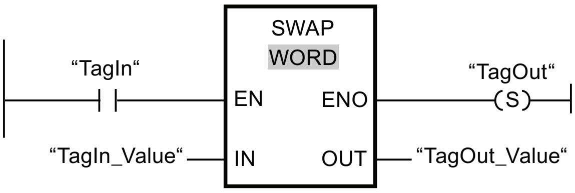
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD, DWORD	I, Q, M, D, L or constant	Operand whose bytes are swapped.
OUT	Output	WORD, DWORD	I, Q, M, D, L	Result

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0000 1111 0101 0101
OUT	TagOut_Value	0101 0101 1111 0000

If operand "TagIn" has the signal state "1", the "Swap" instruction is executed. The order of the bytes is changed and stored in operand "TagOut_Value". If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

Inserting additional inputs and outputs in LAD elements (Page 927)

Conversion operations

CONVERT: Convert value

Description

The "Convert value" instruction reads the content of the IN parameter and converts it according to the data types selected in the instruction box. The converted value is sent to the OUT output.

For information on the possible conversions, refer to the "Auto-Hotspot" section.

The execution of the "Convert value" instruction is only started when the signal state at the EN enable input is "1". If no errors occur during execution, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- Errors such as an overflow occur during execution.
- An operand of data type BYTE, WORD, or DWORD is specified at the IN input. This operand's most significant bit is set. A signed integer (SINT, INT, DINT) is specified at the OUT output. It has the same bit length as the operand at the IN input.

Parameters

The following table shows the parameters of the "Convert value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers, floating-point numbers, CHAR, BCD16, BCD32	I, Q, M, D, L or constant	Value to be converted.
OUT	Output	Bit strings, integers, floating-point numbers, CHAR, BCD16, BCD32	I, Q, M, D, L	Result of the conversion

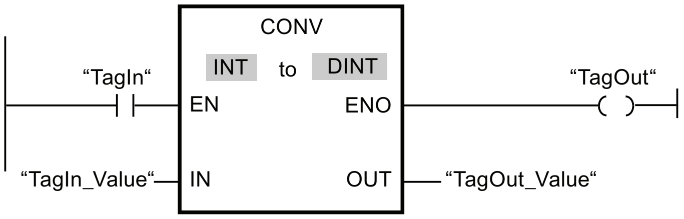
You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

Bit strings (BYTE, WORD, DWORD) cannot be selected in the instruction box. If you enter an operand of data type BYTE, WORD, or DWORD for a parameter of the instruction, the value of the operand is interpreted as an unsigned integer with the same bit length. In this case, data type BYTE is interpreted as USINT, WORD as UINT, and DWORD as UDINT.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the content of operand "TagIn_Value" is read and converted to an integer (32-bit). The result is stored in operand "TagOut_Value". The "TagOut" output is set to "1" if the instruction is executed without errors.

See also

Overview of the valid data types (Page 741)

Explicit conversion of CHAR (Page 816)

Explicit conversion (Page 799)

ROUND: Round numerical value

Description

The "Round numerical value" instruction rounds the value at the IN input to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts this to an integer of data type DINT. If the input value is exactly between an even and odd number, the even number is selected. The result of the instruction is sent to the OUT output and can be queried there.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is processed without errors, the ENO enable output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- Errors such as an overflow occur during execution.

Parameters

The following table shows the parameters of the "Round numerical value" instruction:

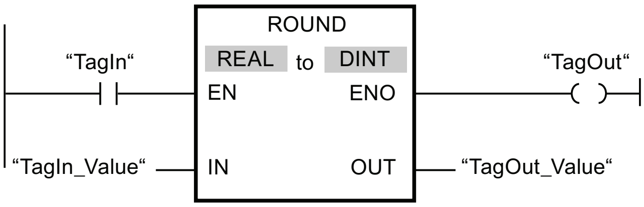
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value to be rounded.
OUT	Output	DINT	I, Q, M, D, L	Result of rounding

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	0	0

If operand "TagIn" has the signal state "1", the "Round numerical value" instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the nearest even integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

CEIL: Generate next higher integer from floating-point number

Description

The "Generate next higher integer from floating-point number" instruction rounds the value at the IN input to the next higher integer. The instruction interprets the value at the IN input as a floating-point number and converts this number to the next higher integer. The result of the instruction is sent to the OUT output and can be queried there. The output value can be greater than or equal to the input value.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is processed without errors, the ENO enable output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- Errors such as an overflow occur during execution.

Parameters

The following table shows the parameters of the "Generate next higher integer from floating-point number" instruction:

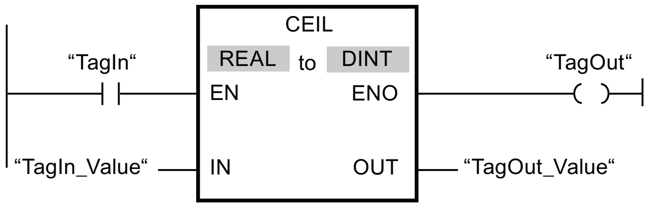
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result with next higher integer

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	1	0

If operand "TagIn" has the signal state "1", the "Generate next higher integer from floating-point number" instruction is executed. The floating-point number at the "TagIn_Value" input is rounded to the next higher integer and sent to the "TagOut_Value" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

FLOOR: Generate next lower integer from floating-point number

Description

The "Generate next lower integer from floating-point number" instruction rounds the value at the IN input to the next lower integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next lower integer. The result of the instruction is sent to the OUT output and can be queried there. The output value can be less than or equal to the input value.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is processed without errors, the ENO enable output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- Errors such as an overflow occur during execution.

Parameters

The following table shows the parameters of the "Generate next lower integer from floating-point number" instruction:

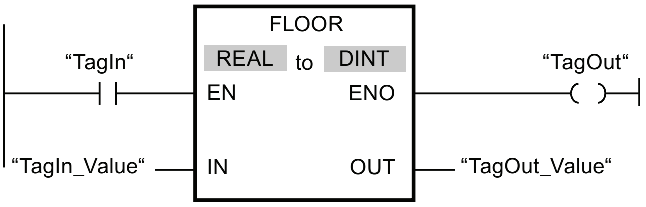
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result with next lower integer

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	0	-1

If operand "TagIn" has the signal state "1", then the "Generate next lower integer from floating-point number" instruction will be executed. The floating-point number at input "TagIn_Value" is rounded to the next lower integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

TRUNC: Truncate numerical value

Description

You can use the "Truncate numerical value" instruction to form an integer from the value at the IN input. The value at the IN input is interpreted as a floating-point number. The instruction selects only the integer part of the floating-point number and sends this to the OUT output without decimal places.

The instruction is only executed if the signal state is "1" at the EN enable input. If the instruction is executed without errors, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- Errors such as an overflow occur during execution.

Parameters

The following table shows the parameters of the "Truncate numerical value" instruction:

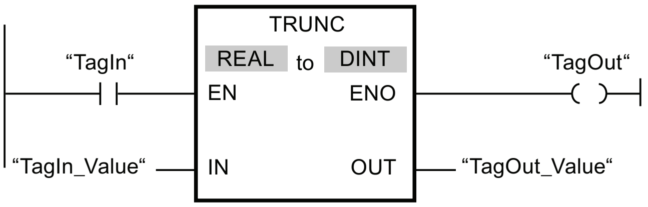
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Integer part of the input value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	- 0.50000000
OUT	TagOut_Value	0	0

If operand "TagIn" has the signal state "1", the "Truncate numerical value" instruction is executed. The integer part of the floating-point number at the "TagIn_Value" input is converted to an integer and sent to the "TagOut_Value" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

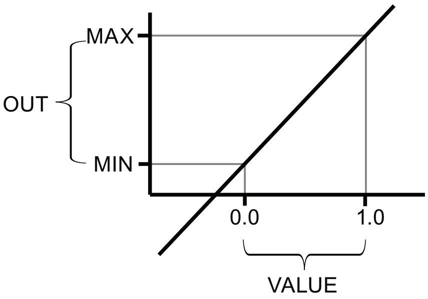
Overview of the valid data types (Page 741)

SCALE_X: Scale

Description

The "Scale" instruction scales the value at input VALUE by mapping it to a specified value range. When the "Scale" instruction is executed, the floating-point value at the VALUE input is scaled to the value range that was defined by the MIN and MAX parameters. The result of the scaling is an integer, which is stored in the OUT output.

The following figure shows an example of how values can be scaled:



The "Scale" instruction can only be executed when the signal state at the EN enable input is "1". In this case, the ENO enable output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The value at the MIN input is greater than or equal to the value at the MAX input.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- An overflow occurs.
- The value at the VALUE input is NaN (Not a Number = result of an invalid arithmetic operation).

Parameters

The following table shows the parameters of the "Scale" instruction:

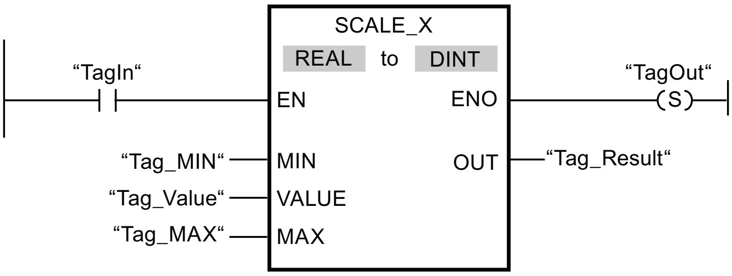
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VALUE	Input	Floating-point numbers	I, Q, M, D, L or constant	Value to be scaled.
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result of scaling

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
MIN	Tag_MIN	10
VALUE	Tag_Value	0.5
MAX	Tag_MAX	30
OUT	Tag_Result	20

If operand "TagIn" has the signal state "1", the "Scale" instruction is executed. The value at the "Tag_Value" input is scaled to the range of values defined by the values at the "Tag_MIN" and "Tag_MAX" inputs. The result is stored in the "Tag_Result" output. If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

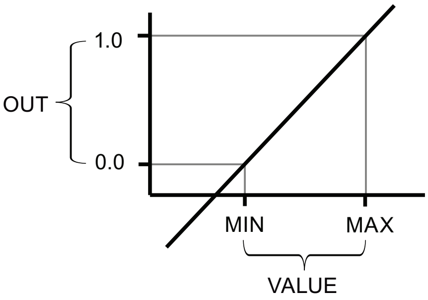
NORM_X: Normalize (Page 1350)

NORM_X: Normalize

Description

The "Normalize" instruction normalizes the value of the tag at input VALUE by mapping it to a linear scale. You can use the MIN and MAX parameters to define the limits of a value range that is applied to the scale. The result at the OUT output is calculated and stored as a floating-point number depending on the location of the value to be normalized within this value range. If the value to be normalized is equal to the value at the MIN input, the OUT output has the value "0.0". If the value to be normalized is equal to the value at the MAX input, the OUT output has the value "1.0".

The following figure shows an example of how values can be normalized:



The "Normalize" instruction can only be executed when the signal state at the EN enable input is "1". In this case, the ENO enable output has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The value at the MIN input is greater than or equal to the value at the MAX input.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- The value at the VALUE input is NaN (result of an invalid arithmetic operation).

Parameters

The following table shows the parameters of the "Normalize" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range

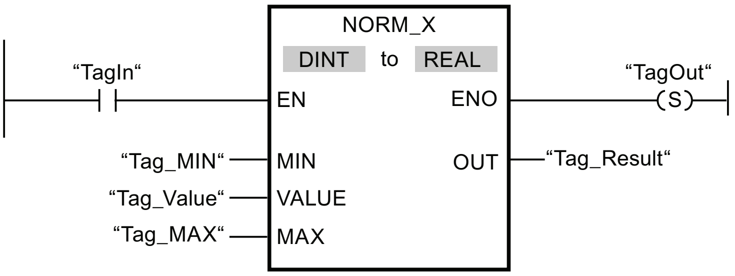
Parameters	Declaration	Data type	Memory area	Description
VALUE	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Value to be normalized.
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
OUT	Output	Floating-point numbers	I, Q, M, D, L	Result of the normalization

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
MIN	Tag_MIN	10
VALUE	Tag_Value	20
MAX	Tag_MAX	30
OUT	Tag_Result	0.5

If operand "TagIn" has the signal state "1", the "Normalize" instruction is executed. The value at the "Tag_Value" input is mapped to the range of values that were defined by the values at the "Tag_MIN" and "Tag_MAX" inputs. The tag value at the "Tag_Value" input is normalized to the defined value range. The result is stored as a floating-point number in the "Tag_Result" output. If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

SCALE_X: Scale (Page 1348)

Program control operations

---(JMP): Jump if RLO = 1

Description

You can use the "Jump if RLO = 1" instruction to interrupt the linear execution of the program and resume it in another network. The destination network must be identified by a jump label (LABEL). The name of this jump label is specified in the placeholder above the instruction.

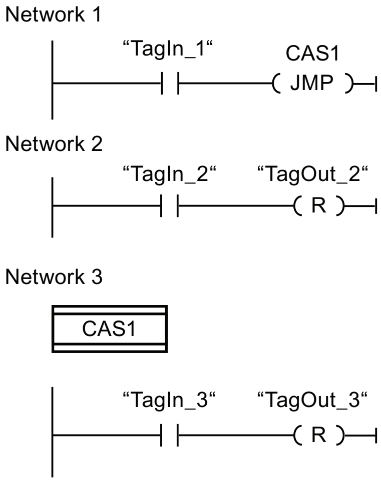
The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block.

If the result of logic operation (RLO) at the input of the instruction is "1", the jump to the network identified by the specified jump label is executed. The jump direction can be towards higher or lower network numbers.

If the condition at the input of the instruction is not fulfilled (RLO = 0), execution of the program continues in the next network.

Example

The following example shows how the instruction works:



If operand "TagIn_1" has the signal state "1", the "Jump if RLO = 1" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has the signal state "1", the "TagOut_3" output is set.

---(JMPN): Jump if RLO = 0

Description

You can use the instruction "Jump if RLO = 0" to interrupt the linear execution of the program and resume it in another network, when the result of logic operation at the input of the instruction is "0". The destination network must be identified by a jump label (LABEL). The name of this jump label is specified in the placeholder above the instruction.

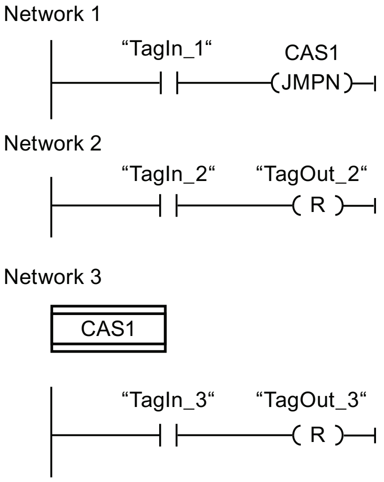
The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block.

If the result of logic operation (RLO) at the input of the instruction is "0", the jump to the network identified by the specified jump label is executed. The jump direction can be towards higher or lower network numbers.

If the result of the logic operation at the input of the instruction is "1", execution of the program continues in the next network.

Example

The following example shows how the instruction works:



If the operand "TagIn_1" has the signal state "0", the instruction "Jump if RLO = 0" is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has the signal state "1", the "TagOut_3" output is set.

See also

Overview of the valid data types (Page 741)

LABEL: Jump label

Description

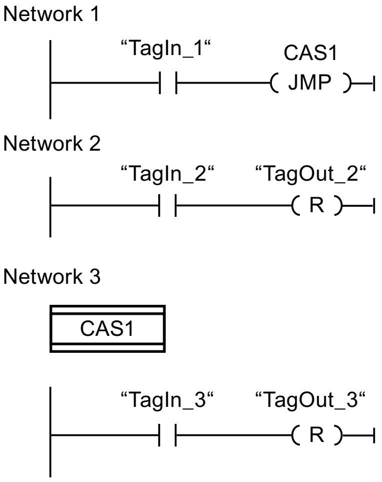
You can use a jump label to identify a destination network, in which the program execution should resume when a jump is executed.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block.

Only one jump label can be placed in a network. Each jump label can jump to several locations.

Example

The following example shows how the instruction works:



If operand "TagIn_1" has the signal state "1", the "Jump if RLO = 1" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has the signal state "1", the "TagOut_3" output is set.

JMP_LIST: Define jump list

Description

The "Define jump list" instruction defines several conditional jumps and resumes the program execution in a specific network depending on the value of the K parameter.

You define the jumps with jump labels (LABEL), which you specify at the outputs of the instruction box. The number of outputs can be expanded in the instruction box. The numbering of the outputs begins with the value "0" and continues in ascending order with each new output. Only jump labels can be specified at the outputs of the instruction. Instructions or operands cannot be specified.

The value of the K parameter specifies the number of the output and thus the jump label where the program execution is to be resumed. If the value in the K parameter is greater than the number of available outputs, the program execution is resumed in the next network of the block.

The "Define jump list" instruction is only executed if the signal state is "1" at the EN enable input.

Parameters

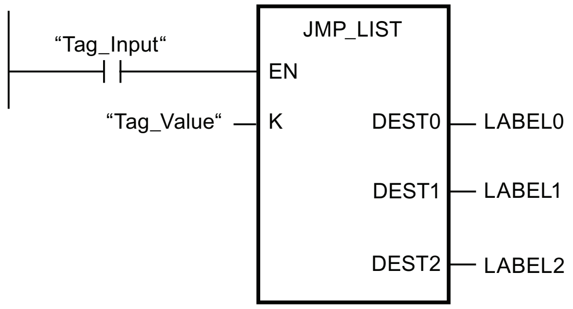
The following table shows the parameters of the "Define jump list" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, L, D	Enable input
K	Input	UINT	I, Q, M, L, D or constant	Specifies the number of the output and thus the jump that is to be made. (K=0 to 99)
DEST0	-	-	-	First jump label
DEST1	-	-	-	Second jump label
Dest(n)	-	-	-	Optional jump labels (n = 2 to 99)

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand/Jump label	Value
K	"Tag_Value"	1
Dest 0	LABEL0	Jump to the network that is identified with the jump label "LABEL0".
Dest 1	LABEL1	Jump to the network that is identified with the jump label "LABEL1".
Dest 2	LABEL2	Jump to the network that is identified with the jump label "LABEL2".

If operand "Tag_Input" has the signal state "1", the "Define jump list" instruction is executed. The program execution is resumed according to the value of operand "Tag_Value" in the network that is identified with the jump label "LABEL1".

See also

Overview of the valid data types (Page 741)

Inserting additional inputs and outputs in LAD elements (Page 927)

Removing inputs and outputs (Page 928)

SWITCH: Jump distributor

Description

You can use the "Jump distributor" instruction to define multiple program jumps to be executed depending on the result of one or more comparison instructions.

You specify the value to be compared in the K parameter. This value is compared with the values that are provided by the various inputs. You can select the comparison method for each individual input. The availability of the various comparison instructions depends on the data type of the instruction.

The following table shows the comparison instructions that are available depending on the selected data type:

Data type	Instruction	Syntax
Bit strings	Equal	==
	Not equal	<>
Integers, floating-point numbers, TIME, DATE, TOD	Equal	==
	Not equal	<>
	Greater or equal	>=
	Less or equal	<=
	Greater than	>
	Less than	<

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box. If you select a comparison instruction and the data type of the instruction is not yet defined, the "<???>" drop-down list only offers the data types that are permitted for the selected comparison instruction.

Execution of the instruction begins with the first comparison and runs until a comparison condition is met. If a comparison condition is met, the subsequent comparison conditions are not considered. If none of the specified comparison conditions are met, the jump at the ELSE output is executed. If no program jump is defined at the ELSE output, execution of the program continues in the next network.

The number of outputs can be expanded in the instruction box. The numbering of the outputs begins with the value "0" and continues in ascending order with each new output. Specify jump labels (LABEL) at the outputs of the instruction. Instructions or operands cannot be specified at the outputs of the instruction.

An input is automatically inserted for each additional output. The jump programmed at an output is executed if the comparison condition of the corresponding input is fulfilled.

Parameters

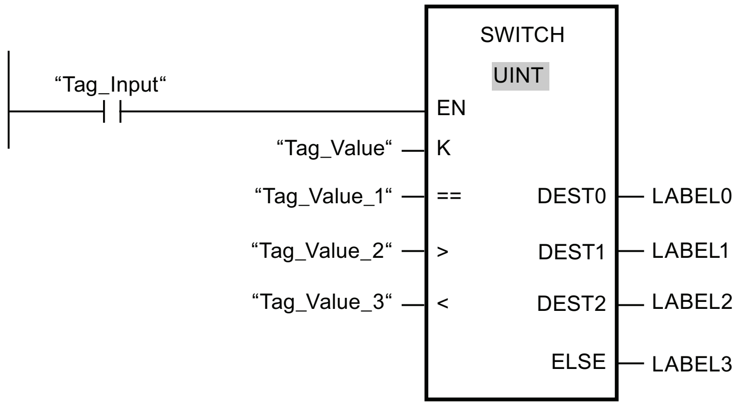
The following table shows the parameters of the "Jump distributor" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
K	Input	UINT	I, Q, M, D, L or constant	Specifies the value to be compared.
<Comparison values>	Input	Bit strings, integers, floating-point numbers, TIME, DATE, TOD	I, Q, M, D, L or constant	Input value with which the value of the K parameter is compared.
DEST0	-	-	-	First jump label
DEST1	-	-	-	Second jump label
DEST(n)	-	-	-	Optional jump labels (n = 2 to 99)
ELSE	-	-	-	Program jump that is executed when none of the comparison conditions are fulfilled.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand/Jump label	Value
K	Tag_Value	23
==	Tag_Value_1	20
>	Tag_Value_2	21
<	Tag_Value_3	19
Dest 0	LABEL0	Jump to jump label "LABEL0", if the value of the K parameter equals 20.
Dest 1	LABEL1	Jump to jump label "LABEL1" if the value of the K parameter is greater than 21.
Dest 2	LABEL2	Jump to jump label "LABEL2", if the value of the K parameter is less than 19.
ELSE	LABEL 3	Jump to jump label "LABEL3", if the none of the comparison conditions are fulfilled.

If the operand "Tag_Input" changes to signal state "1", the instruction "Jump distributor" is executed. The execution of the program is continued in the network that is identified with the jump label "LABEL1".

See also

Overview of the valid data types (Page 741)

Inserting additional inputs and outputs in LAD elements (Page 927)

Removing inputs and outputs (Page 928)

--(RET): Return

Description

You can use the "Return" instruction to stop the execution of a block. The results is three types through which the block processing can be completed.

- Without call of the "Return" instruction

  The block is exited after execution of the last network. The ENO of the function call is set to the signal state "1".
- Call of the "Return" instruction with preceding logic operation (see example)

  If the left connector has the signal state "1", the block will be exited. The ENO of the function call corresponds to the operand.
- Call of the "Return" instruction without previous logic operation

  The block is exited. The ENO of the function call corresponds to the operand.

Note

Only one jumping coil may be used in a network ("Return", "Jump if RLO=1", "Jump if RLO=0").

If the result of logic operation (RLO) at the input of the "Return" instruction is "1", program execution is terminated in the currently called block and resumed after the call function in the calling block (for example, in the calling OB). The status (ENO) of the call function is determined by the parameter of the instruction. This can assume the following values:

- RLO
- TRUE/FALSE
- <Operand>

To set the parameter values, double-click the instruction and select the corresponding value in the drop-down list.

The following table shows the status of the call function if the "Return" instruction is programmed in a network within the called block:

RLO	Parameter value	ENO of the call function
1	RLO	1
	TRUE	1
	FALSE	0
	<Operand>	<Operand>
0	RLO	The program execution continues in the next network of the called block.
	TRUE	
	FALSE	
	<Operand>	

If an OB is completed, another block will be selected and started or executed by the priority class system:

- If the program cycle OB was completed, it will be restarted.
- If an OB is completed that has interrupted another block (e.g. an Alarm OB), then the interrupted block (e.g. program cycle OB) will be executed.

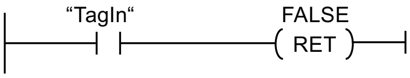
Parameters

The following table shows the parameters of the "Return" instruction:

Parameters	Declaration	Data type	Memory area	Description
RLO	-	-		The status of the call function is set to the signal state of the RLO.
TRUE	-	-		If RLO=1, the status of the call function is set to "1".
FALSE	-	-		If RLO=1, the status of the call function is set to "0".
<Operand>	Input	BOOL	I, Q, M, D, L	If RLO=1, the status of the call function is set to the signal state of the specified operand.

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state"1", the "Return" instruction is executed. Program execution is terminated in the called block and continues in the calling block. The ENO output of the call function is reset to signal state "0".

See also

Overview of the valid data types (Page 741)

Runtime control

RE_TRIGR: Restart cycle monitoring time

Description

You can use the "Restart cycle monitoring time" instruction to restart the cycle monitoring of the CPU. The cycle time monitoring then starts over again for the duration you have set in the CPU configuration. By restarting the cycle monitoring time, you can prevent errors being triggered or the CPU changing to STOP.

The "Restart cycle monitoring time" instruction can be used in blocks of priority class 1 (the cyclic OB) and in the blocks called within it.

If the instruction is called in a block with a higher priority, such as a hardware interrupt, diagnostic interrupt, or cyclic interrupt, the instruction is not executed and the ENO enable output is set to signal state "0".

Parameters

The "Restart cycle monitoring time" instruction has no parameters.

STP: Exit program

Description

You use the "Exit program" instruction to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

When the RLO at the input of the instruction is "1", the CPU changes to STOP mode and program execution is terminated. The signal state at the output of the instruction is not evaluated.

When the RLO is "0" at the input of the instruction, then the instruction will not be executed.

Parameters

The "Exit program" instruction has no parameters.

GetError: Get error locally

Description

The "Get error locally" instruction is used to query the occurrence of errors within a block. If the system signals errors during block execution, detailed information about the first error that occurred is saved in the operand at the ERROR output.

Only operands of the "ErrorStruct" system data type can be specified at the ERROR output. The "ErrorStruct" system data type specifies the exact structure in which the information about the error is stored. Using additional instructions, you can evaluate this structure and program an appropriate response. When the first error has been eliminated, the instruction outputs information about the next error that occurred.

Parameters

The following table shows the parameters of the "Get error locally" instruction:

Parameters	Declaration	Data type	Memory area	Description
ERROR	Output	ErrorStruct	D, L	Error information

Data type "ErrorStuct"

The following table shows the structure of the "ErrorStruct" data type:

Structure components		Data type	Description
ERROR_ID		WORD	Error ID
FLAGS		BYTE	Shows if an error occurred during a block call. 16#01: Error during a block call. 16#00: No error during a block call.
REACTION		BYTE	Default reaction: 0: Ignore (write error), 1: Continue with substitute value "0" (read error), 2: Skip instruction (system error)
CODE_ADDRESS		CREF	Information about the address and type of block
	BLOCK_TYPE	BYTE	Type of block where the error occurred: 1: OB 2: FC 3: FB
	CB_NUMBER	UINT	Number of the code block
	OFFSET	UDINT	Reference to the internal memory
MODE		BYTE	Access mode: Depending on the type of access, the following information can be output: Mode | (A) | (B) | (C) | (D) | (E)

Structure components		Data type	Description					
			0					
			1					Offset
			2			Area		
			3	Location	Scope		Number	
			4			Area		Offset
			5			Area	DB no.	Offset
			6	PtrNo./Acc		Area	DB no.	Offset
			7	PtrNo./Acc	Slot No. / Scope	Area	DB no.	Offset
OPERAND_NUMBER		UINT	Operand number of the machine command					
POINTER_NUMBER_ LOCATION		UINT	(A) Internal pointer					
SLOT_NUMBER_SCOPE		UINT	(B) Storage area in internal memory					
DATA_ADDRESS		NREF	Information about the address of an operand					
	AREA	BYTE	(C) Memory area: L: 16#40 – 4E, 86, 87, 8E, 8F, C0 – CE E: 16#81 A: 16#82 M: 16#83 DB: 16#84, 85, 8A, 8B					
	DB_NUMBER	UINT	(D) Number of the data block					
	OFFSET	UDINT	(E) Relative address of the operand					

Structure component "ERROR_ID"

The following table shows the values that can be output on the structure component "ERROR_ID":

ID (hexadecimal)	ID (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2505	9477	Calling the instruction "Stop" (SFC46) in the user program
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment

ID (hexadecimal)	ID (decimal)	Description
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2534	9524	Block number error FC
2535	9525	Block number error FB
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2942	10562	Read error: Input
2943	10563	Write error: Output

The ENO enable output of the "Get error locally" instruction is only set if the EN enable input has the signal state "1" and error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error locally" instruction.

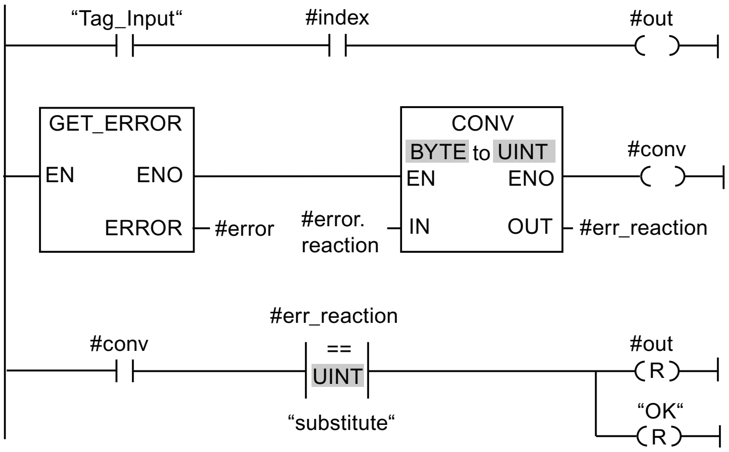
The "Get error locally" instruction can also be used to forward an alarm about the error status to the calling block. To do this, the instruction must be positioned in the last network of the called block.

Note

The "Get error locally" instruction enables local error handling within a block. If "Get error locally" is inserted in the program code of a block, any predefined system responses are ignored when an error occurs.

Example

The following example shows how the instruction works:



When an error occurs, the "Get error locally" instruction returns the error information to the locally created "#error" structure at the ERROR output. The error information is converted and evaluated using the "Equal" comparison instruction. Information about the type of error is the first comparison value assigned to the instruction. The value "1" is specified in operand "substitute" as the second comparison value. If the error is a read error, the condition of the comparison instruction is fulfilled. The "#out" and "OK" outputs are reset in this case.

See also

Overview of the valid data types (Page 741)

Basics of error handling (Page 1009)

Principles of local error handling (Page 1011)

Error output priorities (Page 1012)

Enabling local error handling for a block (Page 1013)

GetErrorID: Get error ID locally

Description

The "Get error ID locally" instruction is used to query the occurrence of errors within a block. If the system signals errors during block execution, the error ID of the first error that occurred is saved in the tag at output ID. Only operands of the WORD data type can be specified at the ID output. When the first error has been eliminated, the instruction outputs the error ID of the next error that occurred.

The output of the "Get error ID locally" instruction is only set if the input of the instruction has the signal state "1" and error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error ID locally" instruction.

The "Get error ID locally" instruction can also be used to forward an alarm about the error status to the calling block. To do this, the instruction must be positioned in the last network of the called block.

Note

The "Get error ID locally" instruction enables local error handling within a block. If the "Get error ID locally" instruction is inserted in the program code of a block, any predefined system responses are ignored when an error occurs.

Parameters

The following table shows the parameters of the "Get error ID locally" instruction:

Parameters	Declaration	Data type	Memory area	Description
ID	Output	WORD	I, Q, M, D, L	Error ID

Parameters ID

The following table shows the values that can be output in the ID parameter:

ID (hexadecimal)	ID (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2505	9477	Calling the instruction "Stop" (SFC46) in the user program
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2534	9524	Block number error FC
2535	9525	Block number error FB
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2942	10562	Read error: Input
2943	10563	Write error: Output

See also

Overview of the valid data types (Page 741)

Basics of error handling (Page 1009)

Principles of local error handling (Page 1011)

Error output priorities (Page 1012)

Enabling local error handling for a block (Page 1013)

Word logic operations

AND: AND logic operation

Description

You can use the "AND logic operation" instruction to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by AND logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are logically ANDed. The result is stored in bit 0 of the OUT output. The same logic operation is executed for all other bits of the specified values.

The number of inputs can be expanded in the instruction box. The added inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are combined with AND logic (ANDed). The result is stored in the OUT output.

The result bit has the signal state "1" only when both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has the signal state "0", the corresponding result bit is reset.

The instruction is only executed if the signal state is "1" at the EN enable input. In this case, the ENO output also has the signal state "1".

If the signal state at the EN enable input is "0", the ENO enable output also has the signal state "0".

Parameters

The following table shows the parameters of the "AND logic operation" instruction:

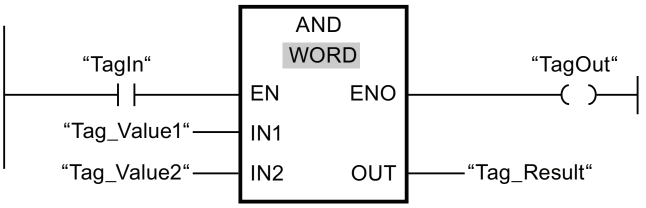
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L or constant	Other inputs whose values are logically combined.
OUT	Output	Bit strings	I, Q, M, D, L	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	01010101 01010101
IN2	Tag_Value2	00000000 00001111
OUT	Tag_Result	00000000 00000101

If operand "TagIn" has the signal state "1", the "AND logic operation" instruction is executed. The value of operand "Tag_Value1" and the value of operand "Tag_Value2" are ANDed. The result is mapped bit-for-bit and output in operand "Tag_Result". The ENO output and the "TagOut" output are set to signal state "1".

See also

Overview of the valid data types (Page 741)

Inserting additional inputs and outputs in LAD elements (Page 927)

Removing inputs and outputs (Page 928)

Basics of the EN/ENO mechanism (Page 819)

OR: OR logic operation

Description

You can use the "OR logic operation" instruction to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by OR logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are logically ORed. The result is stored in bit 0 of the OUT output. The same logic operation is executed for all bits of the specified tags.

The number of inputs can be expanded in the instruction box. The added inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are combined with OR logic (ORed). The result is stored in the OUT output.

The result bit has the signal state "1" when at least one of the two bits in the logic operation has the signal state "1". If both of the bits of the logic operation have signal state "0", the corresponding result bit is reset.

The instruction is only executed if the signal state is "1" at the EN enable input. In this case, the ENO output also has the signal state "1".

If the signal state at the EN enable input is "0", the ENO enable output also has the signal state "0".

Parameters

The following table shows the parameters of the "OR logic operation" instruction:

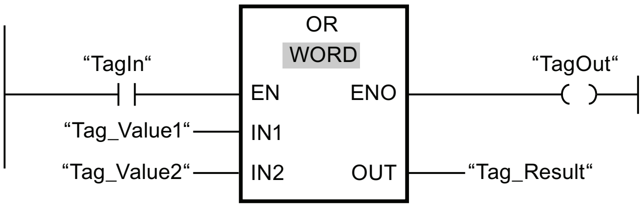
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L or constant	Other inputs whose values are logically combined.
OUT	Output	Bit strings	I, Q, M, D, L	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	01010101 01010101
IN2	Tag_Value2	00000000 00001111
OUT	Tag_Result	01010101 01011111

If operand "TagIn" has the signal state "1", the "OR logic operation" instruction is executed. The value of operand "Tag_Value1" and the value of operation "Tag_Value2" are ORed. The result is mapped bit-for-bit and output in operand "Tag_Result". The ENO output and the "TagOut" output are set to signal state "1".

See also

Overview of the valid data types (Page 741)

Inserting additional inputs and outputs in LAD elements (Page 927)

Removing inputs and outputs (Page 928)

Basics of the EN/ENO mechanism (Page 819)

XOR: EXCLUSIVE OR logic operation

Description

You can use the "EXCLUSIVE OR logic operation" to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by EXCLUSIVE OR logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are logically EXCLUSIVELY ORed. The result is stored in bit 0 of the OUT output. The same logic operation is executed for all other bits of the specified value.

The number of inputs can be expanded in the instruction box. The added inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are combined with EXCLUSIVE OR logic (EXCLUSIVELY ORed). The result is stored in the OUT output.

The result bit has the signal state "1" when one of the two bits in the logic operation has the signal state "1". If both of the bits of the logic operation have signal state "1" or "0", the corresponding result bit is reset.

The instruction is only executed if the signal state is "1" at the EN enable input. In this case, the ENO output also has the signal state "1".

If the signal state at the EN enable input is "0", the ENO enable output also has the signal state "0".

Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation" instruction:

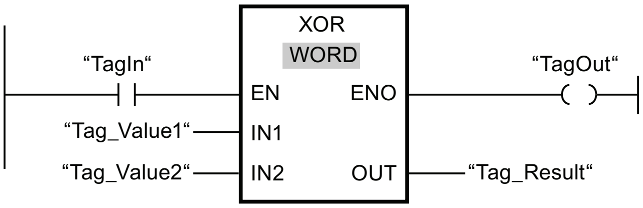
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L or constant	Other inputs whose values are logically combined.
OUT	Output	Bit strings	I, Q, M, D, L	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	01010101 01010101
IN2	Tag_Value2	00000000 00001111
OUT	Tag_Result	01010101 01011010

If operand "TagIn" has the signal state "1", the "EXCLUSIVE OR logic operation" instruction is executed. The value of operand "Tag_Value1" and the value of operand "Tag_Value2" are EXCLUSIVELY ORed. The result is mapped bit-for-bit and output in operand "Tag_Result". The ENO output and the "TagOut" output are set to signal state "1".

See also

Overview of the valid data types (Page 741)

Inserting additional inputs and outputs in LAD elements (Page 927)

Removing inputs and outputs (Page 928)

Basics of the EN/ENO mechanism (Page 819)

INV: Create ones complement

Description

You can use the "Create ones complement" instruction to invert the signal state of the bits at the IN input. When the instruction is processed, the value at the IN input and a hexadecimal template (W#16#FFFF for 16-bit numbers or DW#16#FFFF FFFF for 32-bit numbers) are logically EXCLUSIVELY ORed. As a result, the signal state of the individual bits is inverted and sent to the OUT output.

The instruction is only executed if the signal state is "1" at the EN enable input. In this case, the ENO output also has the signal state "1".

If the signal state at the EN enable input is "0", the ENO enable output also has the signal state "0".

Parameters

The following table shows the parameters of the "Create ones complement" instruction:

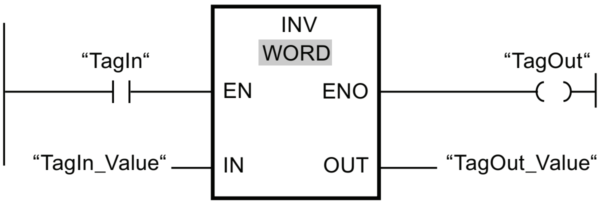
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	I, Q, M, D, L or constant	Input value
OUT	Output	Bit strings, integers	I, Q, M, D, L	Ones complement of the value at input IN

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	W#16#000F	W#16#7E
OUT	TagOut_Value	W#16#FFF0	W#16#81

If operand "TagIn" has the signal state "1", then the "Create ones complement" instruction will be executed. The instruction inverts the signal state of the individual bits at the TagIn_Value" input and writes the result to the "TagOut_Value" output. The ENO output and the "TagOut" output are set to signal state "1".

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

DECO: Decode

Description

You can use the "Decode" instruction to set a bit in the output value specified by the input value.

The "Decode" instruction reads the value at the IN input and sets the bit in the output value whose bit position corresponds to the read value. The other bits in the output value will be overwritten with zeroes. When the value at the IN input is greater than 31, a modulo 32 instruction is executed.

The "Decode" instruction is only started when the signal state at the EN enable input is "1". If no errors occur during execution, the ENO output also has the signal state "1".

If the signal state at the EN enable input is "0", the ENO enable output also has the signal state "0".

Parameters

The following table shows the parameters of the "Decode" instruction:

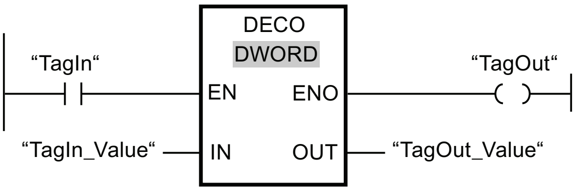
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	UINT	I, Q, M, D, L or constant	Input value
OUT	Output	Bit strings	I, Q, M, D, L	Output value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

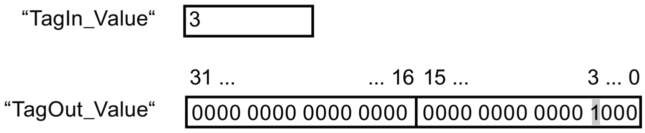
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following figure shows how the instruction works using specific operand values:



If operand "TagIn" has the signal state "1", the "Decode" instruction is executed. The instruction reads bit number "3" from the value at the "TagIn_Value" input and sets the third bit in the value at the "TagOut_Value" output.

If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

ENCO: Encode

Description

You can use the "Encode" instruction to read the bit number of the least significant bit in the input value and to send it to the OUT output.

The "Encode" instruction selects the least significant bit of the value at the IN input and writes its bit number to the tag in the OUT output.

The "Encode" instruction is only started when the signal state at the EN enable input is "1". If no errors occur during execution, the ENO output also has the signal state "1".

If the signal state at the EN enable input is "0", the ENO enable output also has the signal state "0".

Parameters

The following table shows the parameters of the "Encode" instruction:

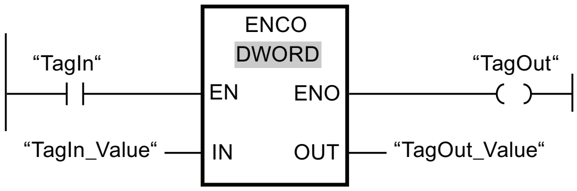
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings	I, Q, M, D, L or constant	Input value
OUT	Output	INT	I, Q, M, D, L	Output value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

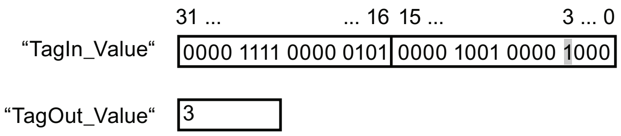
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following figure shows how the instruction works using specific operand values:



If operand "TagIn" has the signal state "1", the "Encode" instruction is executed. The instruction selects the least significant bit at the "TagIn_Value" input and writes bit position "3" to the tag in the "TagOut_Value" output.

If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

SEL: Select

Description

Depending on a switch (G input), the "Select" instruction selects one of the IN0 or IN1 inputs and copies its content to the OUT output. When the G input has the signal state "0", the value at the IN0 input is moved. When the G input has the signal state "1", the value at the IN1 input is copied to the OUT output.

Execution of the instruction assumes the signal state "1" at the EN enable input and that the tags are of the same data type at all parameters. If the instruction is executed without errors, the ENO enable output also has the signal state "1".

The ENO enable output is reset when the EN enable input has the signal state "0" or errors occur during execution of the instruction.

Parameters

The following table shows the parameters of the "Select" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
G	Input	BOOL	I, Q, M, D, L	Switch
IN0	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, characters	I, Q, M, D, L or constant	First input value

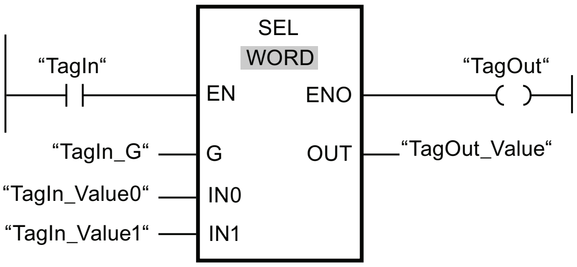
Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, characters	I, Q, M, D, L or constant	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, characters	I, Q, M, D, L	Result

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
G	TagIn_G	0	1
IN0	TagIn_Value0	W#16#0000	W#16#4C
IN1	TagIn_Value1	W#16#FFFF	W#16#5E
OUT	TagOut_Value	W#16#0000	W#16#5E

If operand "TagIn" has the signal state "1", the "Select" instruction is executed. Based on the signal state at the "TagIn_G" input, the value at the "TagIn_Value0" or "TagIn_Value1" input is selected and copied to the "TagOut_Value" output. If the instruction is executed without errors, enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

MUX: Multiplex

Description

You can use the "Multiplex" instruction to copy the content of a selected input to the OUT output. The number of selectable inputs of the instruction box can be expanded. The inputs are automatically numbered in the box. Numbering starts at IN0 and continues consecutively with each new input. You use the K parameter to define the input whose content is to be copied to the OUT output. If the value of the K parameter is greater than the number of available inputs, the content of the ELSE parameter is copied to the OUT output and the ENO enable output is assigned the signal state "0".

The "Multiplex" instruction can only be executed, when the tags in all inputs and in the OUT output have the same data type. The K parameter is an exception, since only integers can be specified for it.

The instruction is only executed if the signal state is "1" at the EN enable input. If no errors occur during execution, the ENO output also has the signal state "1".

The ENO enable output is reset if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value of the K parameter is greater than the number of available inputs.
- Errors occurred during execution of the instruction.

Parameters

The following table shows the parameters of the "Multiplex" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
K	Input	UINT	I, Q, M, D, L or constant	Specifies the input whose content is to be copied.
IN0	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, CHAR	I, Q, M, D, L or constant	First input value
IN1	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, CHAR	I, Q, M, D, L or constant	Second input value
INn	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, CHAR	I, Q, M, D, L or constant	Optional input values

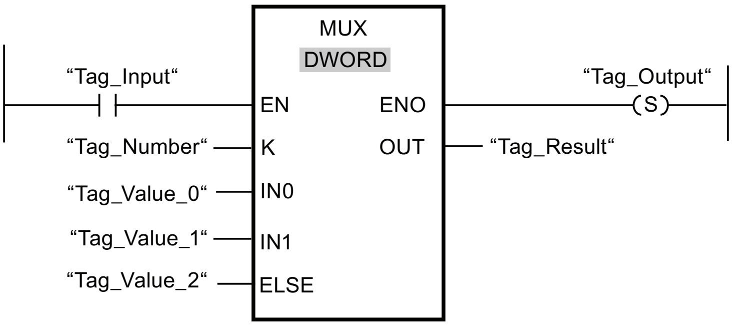
Parameters	Declaration	Data type	Memory area	Description
ELSE	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, CHAR	I, Q, M, D, L or constant	Specifies the value to be copied with K > n.
OUT	Output	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, CHAR	I, Q, M, D, L	Output to which the value is to be copied.

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
K	Tag_Number	1
IN0	Tag_Value_0	DW#16#00000000
IN1	Tag_Value_1	DW#16#3E4A7D
ELSE	Tag_Value_2	DW#16#FFFF0000
OUT	Tag_Result	DW#16#3E4A7D

If operand "Tag_Input" has the signal state "1", the "Multiplex" instruction is executed. Depending in the value of operand Tag_Number, the value at the "Tag_Value_1" input is copied and assigned to the operand at the "Tag_Result" output. If the instruction is executed without errors, the "ENO" and Tag_Output outputs are set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

DEMUX: Demultiplex

Description

You can use the "Demultiplex" instruction to copy the content of the IN input to a selected output. The number of selectable outputs can be expanded in the instruction box. The outputs are automatically numbered in the box. Numbering starts at OUT0 and continues consecutively with each new input. You use the K parameter to define the output to which the content of the IN input is to be copied. The other outputs are not changed. If the value of the K parameter is greater than the number of available outputs, then the content of the IN input will be copied to the ELSE parameter and the signal state "0" is assigned to the ENO enable output.

The "Demultiplex" instruction can only be executed if the tags in the IN input and in all outputs have the same data type. The K parameter is an exception, since only integers can be specified for it.

The instruction is only executed if the signal state is "1" at the EN enable input. If no errors occur during execution, the ENO output also has the signal state "1".

The ENO enable output is reset if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value of the K parameter is greater than the number of available outputs.
- Errors occurred during execution of the instruction.

Parameters

The following table shows the parameters of the "Demultiplex" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
K	Input	UINT	I, Q, M, D, L or constant	Specifies the output to which the input value (IN) is copied.
IN	Input	Bit strings, integers, floating-point numbers, CHAR, TIME	I, Q, M, D, L or constant	Input value
OUT0	Output	Bit strings, integers, floating-point numbers, CHAR, TIME	I, Q, M, D, L	First output

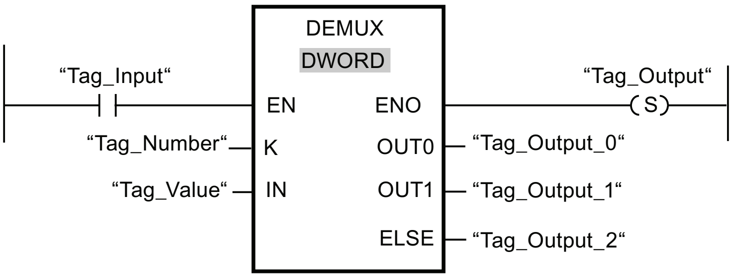
Parameters	Declaration	Data type	Memory area	Description
OUT1	Output	Bit strings, integers, floating-point numbers, CHAR, TIME	I, Q, M, D, L	Second output
OUTn	Output	Bit strings, integers, floating-point numbers, CHAR, TIME	I, Q, M, D, L	Optional outputs
ELSE	Output	Bit strings, integers, floating-point numbers, CHAR, TIME	I, Q, M, D, L	Output to which the input value (IN) is copied when K > n.

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on available data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Table 9- 22 Input values of the "Demultiplex" instruction before network execution

Parameters	Operand	Values	
K	Tag_Number	1	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#3E4A7D

Table 9- 23 Output values of the "Demultiplex" instruction after network execution

Parameters	Operand	Values	
OUT0	Tag_Output_0	Unchanged	Unchanged
OUT1	Tag_Output_1	DW#16#FFFFFFFF	Unchanged
ELSE	Tag_Output_2	Unchanged	DW#16#3E4A7D

If the "Tag_Input" input has the signal state "1", the "Demultiplex" instruction is executed. Depending on the value of operand "Tag_Number", the value at the IN input is copied to the corresponding output.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

Shift and rotate

SHR: Shift right

Description

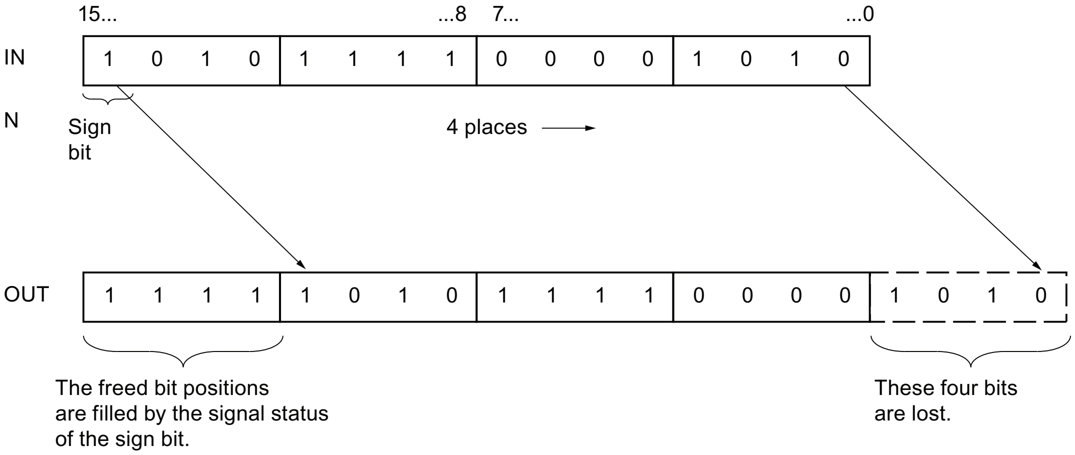
You can use the "Shift right" instruction to shift the content of the operand at the IN input bit-by-bit to the right and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be shifted.

When the value at the N parameter is "0", the value at the IN input is copied to the operand at the OUT output.

When the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is shifted by the available number of bit positions to the right.

In the case of unsigned values, the freed bit positions in the left area of the operand are filled with zeroes when shifting occurs. If the specified value has a sign, the free bit positions are filled with the signal state of the sign bit.

The following figure show how the content of an operand of integer data type is shifted four bit positions to the right:



The "Shift right" instruction is only executed if the signal state is "1" at the EN enable input. In this case, the ENO enable output also has the signal state "1".

If the signal state at the EN enable input is "0", the ENO enable output also has the signal state "0".

Parameters

The following table shows the parameters of the "Shift right" instruction:

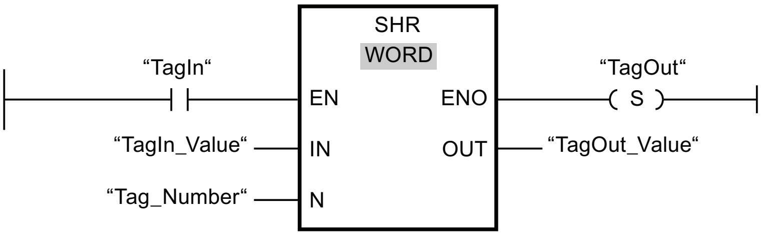
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	I, Q, M, D, L or constant	Value to be shifted.
N	Input	UINT	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted.
OUT	Output	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	3
OUT	TagOut_Value	0000 0111 1111 0101

If operand "TagIn" has the signal state "1", the "Shift right" instruction is executed. The content of operand "TagIn_Value" is shifted three bit positions to the right. The result is sent to the "TagOut_Value" output. If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

SHL: Shift left

Description

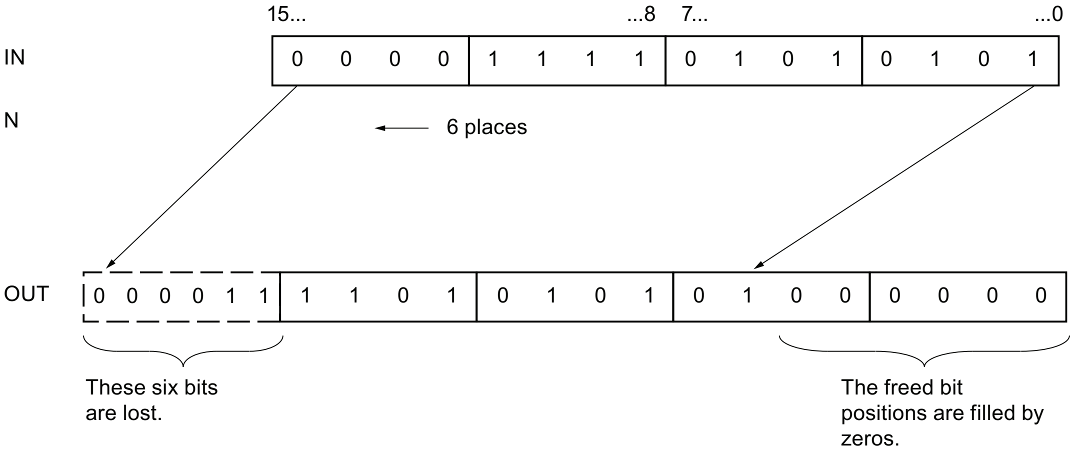
You can use the "Shift left " instruction to shift the content of the operand at the IN input bit-by-bit to the left and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be shifted.

When the value at the N parameter is "0", the value at the IN input is copied to the operand at the OUT output.

When the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is shifted by the available number of bit positions to the left.

The bit positions in the right part of the operand freed by shifting are filled with zeros.

The following figure shows how the content of an operand of WORD data type is shifted six bit positions to the left:



The "Shift left" instruction is only executed if the signal state is "1" at the EN enable input. In this case, the ENO enable output also has the signal state "1".

If the signal state at the EN enable input is "0", the ENO enable output also has the signal state "0".

Parameters

The following table shows the parameters of the "Shift left" instruction:

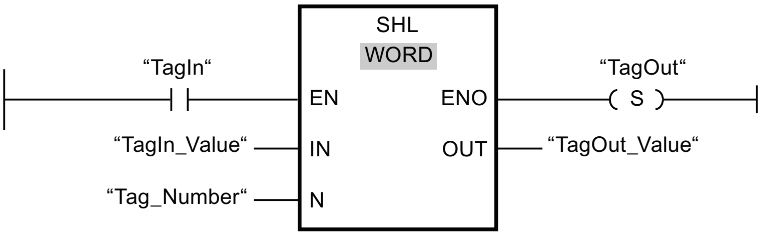
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	I, Q, M, D, L or constant	Value to be shifted.
N	Input	UINT	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted.
OUT	Output	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	4
OUT	TagOut_Value	1111 1010 1111 0000

If operand "TagIn" has the signal state "1", the "Shift left" instruction is executed. The content of operand "TagIn_Value" is shifted four bit positions to the left. The result is sent to the "TagOut_Value" output. If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

ROR: Rotate right

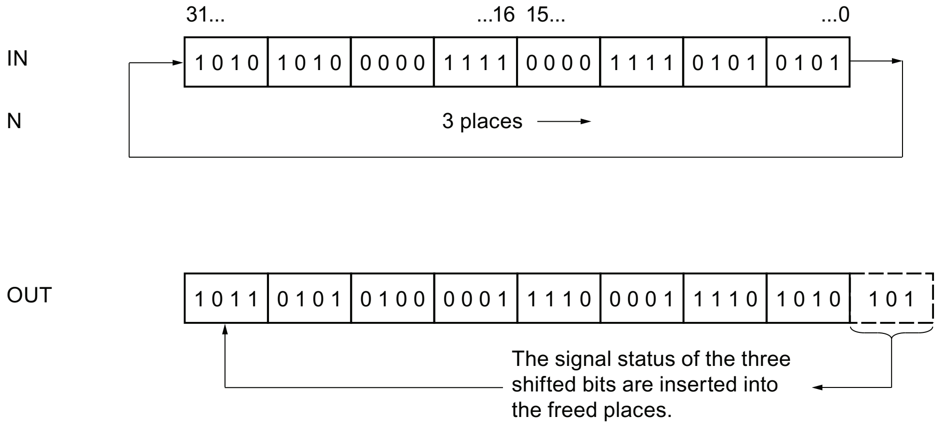
Description

The "Rotate right" instruction rotates the content of the operand at the IN input bit-by-bit to the right and queries the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

When the value at the N parameter is "0", the value at the IN input is copied to the operand at the OUT output.

When the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of DWORD data type is rotated three positions to the right:



The "Rotate right" instruction is only executed if the signal state is "1" at the EN enable input. In this case, the ENO enable output also has the signal state "1".

If the signal state at the EN enable input is "0", the ENO enable output also has the signal state "0".

Parameters

The following table shows the parameters of the "Rotate right" instruction:

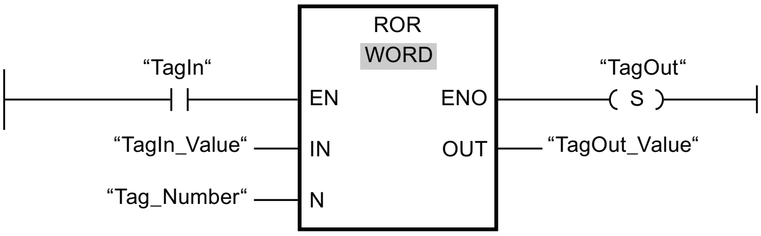
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings	I, Q, M, D, L or constant	Value to be rotated.
N	Input	UINT	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated.
OUT	Output	Bit strings	I, Q, M, D, L	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0000 1111 1001 0101
N	Tag_Number	5
OUT	TagOut_Value	1010 1000 0111 1100

If operand "TagIn" has the signal state "1", the "Rotate right" instruction is executed. The content of operand "TagIn_Value" is rotated five bit positions to the right. The result is sent to the "TagOut_Value" output. If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

ROL: Rotate left

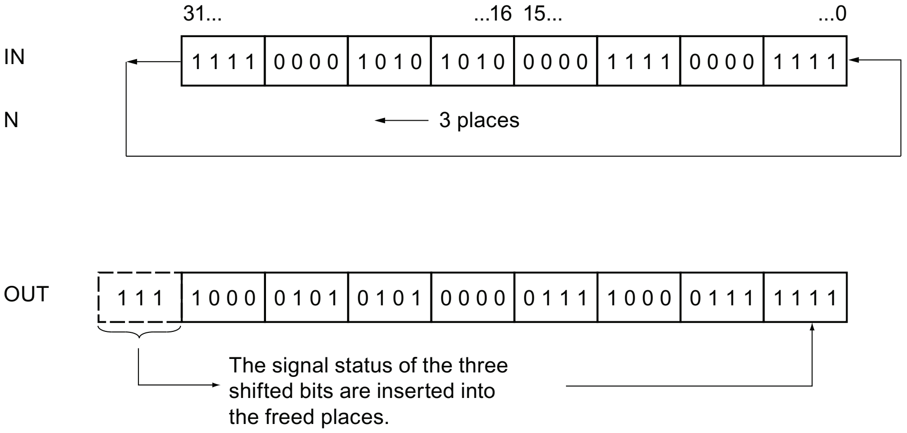
Description

The "Rotate left" instruction rotates the content of the operand at the IN input bit-by-bit to the left and queries the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

When the value at the N parameter is "0", the value at the IN input is copied to the operand at the OUT output.

When the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of DWORD data type is rotated three positions to the left:



The "Rotate left" instruction is only executed if the signal state is "1" at the EN enable input. In this case, the ENO enable output also has the signal state "1".

If the signal state at the EN enable input is "0", the ENO enable output also has the signal state "0".

Parameters

The following table shows the parameters of the "Rotate left" instruction:

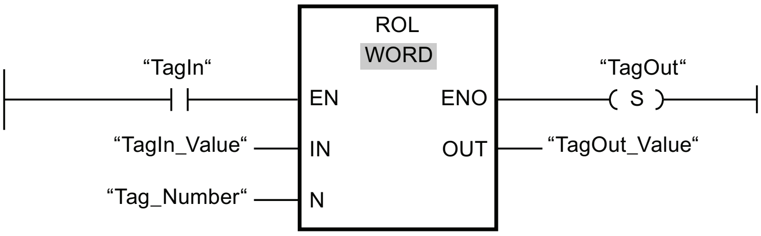
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings	I, Q, M, D, L or constant	Value to be rotated.
N	Input	UINT	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated.
OUT	Output	Bit strings	I, Q, M, D, L	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	1010 1000 1111 0110
N	Tag_Number	5
OUT	TagOut_Value	0001 1110 1101 0101

If the "TagIn" input has the signal state "1", the "Rotate left" instruction is executed. The content of operand "TagIn_Value" is rotated five bit positions to the left. The result is sent to the "TagOut_Value" output. If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

9.8.2.2 FBD

Bit logic operations

&: AND logic operation

Description

You can use the instruction "AND logic operation" to query the signal states of two or more specified operands and evaluate them according to the AND truth table.

If the signal state of all the operands is "1", then the condition is fulfilled and the instruction returns the result "1". If the signal state of one of the operands is "0", then the condition is not fulfilled and the instruction generates the result "0".

If the instruction "AND logic operation" is the first instruction in a logic string, it saves the result of its signal state query in the RLO bit.

Each "AND logic operation" instruction that is not the first operation in the logic sequence, logically combines the result of its signal state query with the value saved in the RLO bit. This logical combination is performed according to the AND truth table.

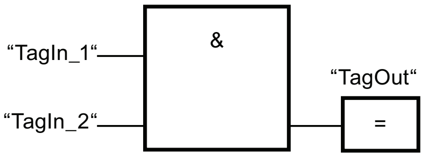
Parameter

The following table shows the parameters of the instruction "AND logic operation":

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



Output "TagOut" is set, when the signal state of the operands "TagIn_1" and "TagIn_2" is "1".

See also

AND truth table (Page 1394)

Example of detecting the direction of a conveyor belt (Page 1212)

Example of controlling room temperature (Page 1217)

Overview of the valid data types (Page 741)

Adding additional inputs and outputs to FBD elements (Page 965)

Insert input (Page 1398)

AND truth table

Results of the logic operation

The following table shows the results that arise from the AND logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	1	1
0	1	0
1	0	0
0	0	0

See also

&: AND logic operation (Page 1393)

>=1: OR logic operation

Description

You can use the instruction "OR logic operation" to query the signal states of two or more specified operands and evaluate them according to the OR truth table.

If the signal state of one of the operands is "1", then the condition is fulfilled and the instruction returns the result "1". If the signal state of all the operands is "0", then the condition is not fulfilled and the instruction generates the result "0".

If the "OR logic operation" instruction is the first instruction in a logic string, it saves the result of its signal state query in the RLO bit.

Each "OR logic operation" instruction that is not the first operation in the logic sequence, logically combines the result of its signal state query with the value saved in the RLO bit. This logical combination is performed according to the OR truth table.

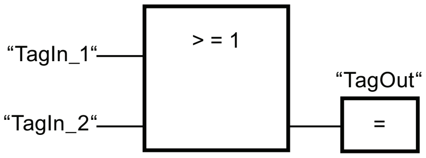
Parameter

The following table shows the parameters of the instruction "OR logic operation":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



Output "TagOut" is set, when the signal state of the operands "TagIn_1" or "TagIn_2" is "1".

See also

OR truth table (Page 1395)

Example of controlling a conveyor belt (Page 1211)

Adding additional inputs and outputs to FBD elements (Page 965)

Overview of the valid data types (Page 741)

Insert input (Page 1398)

OR truth table

Results of the logic operation

The following table shows the results that arise from the OR logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	0	1
0	1	1
1	1	1
0	0	0

See also

>=1: OR logic operation (Page 1394)

X: EXCLUSIVE OR logic operation

Description

You can use the "EXCLUSIVE OR logic operation" instruction to query the result of a signal state query according to the EXCLUSIVE OR truth table.

With an "EXCLUSIVE OR logic operation" instruction, the signal state is "1" when the signal state of one of the two specified operands is "1". When more than two operands are queried, the common RLO is "1" if an odd number of the queried operands returns the result "1".

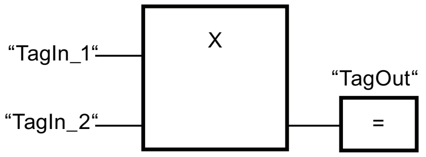
Parameter

The following table shows the parameters of the instruction "EXCLUSIVE OR logic operation":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the signal state of the operands "TagIn_1" and "TagIn_2" is "1". When both operands return the signal state "1" or "0", the output "TagOut" is reset.

See also

EXCLUSIVE OR truth table (Page 1397)

Adding additional inputs and outputs to FBD elements (Page 965)

Overview of the valid data types (Page 741)

Insert input (Page 1398)

EXCLUSIVE OR truth table

Results of the logic operation

The following table shows the results that arise from the EXCLUSIVE OR logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	0	1
0	1	1
1	1	0
0	0	0

The following table shows the results that arise from the EXCLUSIVE OR logic operation of three operands:

Signal state of the first operand	Signal state of the second operand	Signal state of the third operand	Result of the logic operation
1	0	0	1
0	1	1	0
0	1	0	1
1	0	1	0
0	0	1	1
1	1	0	0
1	1	1	1
0	0	0	0

See also

X: EXCLUSIVE OR logic operation (Page 1396)

Insert input

Description

The "Insert input" instruction is used to add an input to the box of one of the following instructions:

- "AND logic operation"
- "OR logic operation"
- "EXCLUSIVE OR logic operation"

You can query the signal state of several operands by the extension of an instruction box.

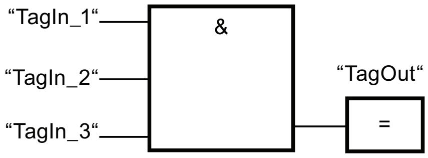
Parameter

The following table shows the parameters of the instruction "Insert input":

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



The box of the "AND logic operation" instruction was extended by an additional input at which the signal state of the operand "TagIn_3" is queried. Output "TagOut" is set, when the signal state of the operands "TagIn_1", "TagIn_2" and "TagIn_3" returns the signal state "1".

See also

&: AND logic operation (Page 1393)

>=1: OR logic operation (Page 1394)

X: EXCLUSIVE OR logic operation (Page 1396)

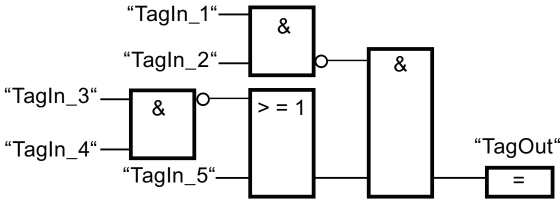
Invert RLO

Description

You use the "Invert RLO" instruction to invert the signal state of the result of logic operation (RLO).

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The input "TagIn_1" and/or "TagIn_2" has signal state "0".
- The input "TagIn_3" and/or "TagIn_4" has signal state "0" or the input "TagIn_5" has signal state "1".

=: Assignment

Description

You can use the "Assignment" instruction to set the bit of a specified operand. If the result of logic operation (RLO) at the box input has the signal state "1", the specified operand is set to signal state "1". If the signal state at the box input is "0", the bit of the specified operand is reset to "0".

The instruction does not influence the RLO. The RLO at the box input is assigned directly to the operand above the assignment box.

The "Assignment" instruction can be placed at any position in the logic string.

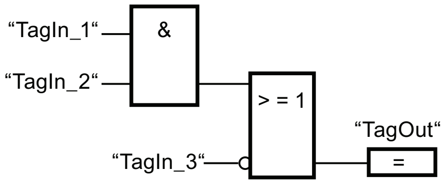
Parameter

The following table shows the parameters of the "Assignment" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:



The operand "TagOut" is set at the output of the "Assignment" instruction when one of the following conditions is fulfilled:

- The inputs "TagIn_1" and "TagIn_2" have the signal state "1".
- The signal state at the input "TagIn_3" is "0".

See also

Overview of the valid data types (Page 741)

Example of detecting the fill level of a storage area (Page 1214)

Example of controlling room temperature (Page 1217)

/=: Negate assignment

Description

The "Negate assignment" instruction inverts the result of logic operation (RLO) and assigns this to the operand above the box. If the RLO at the input of the box is "1", the binary operand is reset. If the RLO at the input of the box is "0", the binary operand is set to signal state "1".

The instruction does not influence the RLO. The RLO at the box input is transferred directly to the box output.

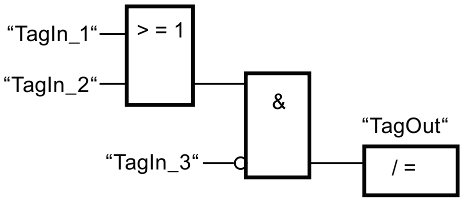
Parameters

The following table shows the parameters of the "Negate assignment" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the negated RLO is assigned.

Example

The following example shows the mode of operation of the "Negate assignment" instruction:



The operand "TagOut" is reset when the following conditions are fulfilled:

- The operand "TagIn_1" or "TagIn_2" has the signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Overview of the valid data types (Page 741)

R: Reset output

Description

You can use the "Reset output" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed if the result of logic operation (RLO) at the box input is "1". If the box input has the signal state "1", the specified operand is reset to "0". If there is an RLO of "0" at the box input, the signal state of the specified operand remains unchanged.

Executing the instruction does not influence the RLO. The RLO at the box input is transferred directly to the box output.

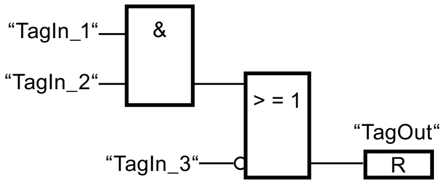
Parameters

The following table shows the parameters of the "Reset output" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand that is reset if RLO = "1".

Example

The following example shows how the instruction works:



Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Example of controlling a conveyor belt (Page 1211)

Example of detecting the direction of a conveyor belt (Page 1212)

Overview of the valid data types (Page 741)

S: Set output

Description

You can use the "Set output" instruction to reset the signal state of a specified operand to "1".

The instruction is only executed if the result of logic operation (RLO) at the box input is "1". If the box input has the signal state "1", the specified operand is set to "1". If there is an RLO of "0" at the box input, the signal state of the specified operand remains unchanged.

Executing the instruction does not influence the RLO. The RLO at the box input is transferred directly to the box output.

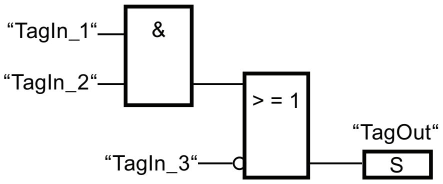
Parameters

The following table shows the parameters of the "Set output" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand that is set when RLO = "1".

Example

The following example shows how the instruction works:



The operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Overview of the valid data types (Page 741)

SET_BF: Set bit field

Description

You use the "Set bit field" instruction to set several bits starting from a certain address.

You use the N input to define the number of bits to be set. The address of the first bit to be set is defined by (<Operand>). If the value of the N input is greater than the number of bits in a selected byte, the bits of the next byte are set. The bits remain set until they are explicitly reset, for example, by another instruction.

The instruction executes only if the result of logic operation (RLO) at the EN input is "1". If the RLO at the EN input is "0", the instruction does not execute.

The "Set bit field" instruction can also be placed without preceding logic operation at the start or end of the logic string.

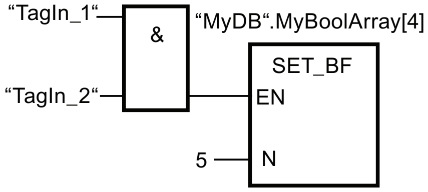
Parameter:

The following table shows the parameters of the "Set bit field" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
N	Input	UINT	Constant	Number of bits to be set
<Operand>	Output	BOOL	I, Q, M With a DB or an IDB, an element of an array [..] of BOOL	Pointer to the first bit to be set.

Example

The following example shows the mode of operation of the "Set bit field" instruction:



If the operands "TagIn_1" and "TagIn_2" have the signal state "1", 5 bits are set starting at the address of the operand "MyDB".MyBoolArray[4].

See also

Overview of the valid data types (Page 741)

RESET_BF: Reset bit field

Description

You use the "Reset bit field" instruction to re set several bits starting from a certain address.

You use the value of the N input to define the number of bits to be reset. The address of the first bit to be reset is defined by (<Operand>). If the value of the N input is greater than the number of bits in a selected byte, the bits of the next byte are reset. The bits remain set until they are explicitly reset, for example, by another instruction.

The instruction executes only if the result of logic operation (RLO) at the EN enable input is "1". If the RLO at the EN enable input is "0", the instruction does not execute.

The "Reset bit field" instruction can also be placed without preceding logic operation at the start or end of the logic string.

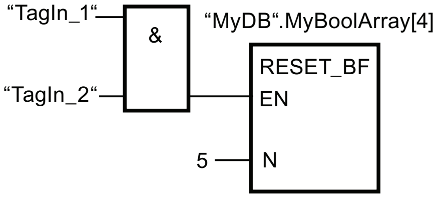
Parameters

The following table shows the parameters of the "Reset bit field" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
N	Input	UINT	Constant	Number of bits to be reset
<Operand>	Output	BOOL	I, Q, M With a DB or an IDB, an element of an array [..] of BOOL	Pointer to the first bit to be reset.

Example

The following example shows the mode of operation of the "Reset bit field" instruction:



If the operands "TagIn_1" and "TagIn_2" have the signal state "1", 5 bits are reset starting at the address of the operand "MyDB".MyBoolArray[4].

See also

Overview of the valid data types (Page 741)

SR: Set/reset flip-flop

Description

The "Set/reset flip-flop" instruction is used to set or reset the bit of a specified operand based on the signal state of the inputs S and R1. If the signal state is "1" at input S and "0" at input R1, the specified operand is set to "1". If the signal state is "0" at input S and "1" at input R1, the specified operand will be reset to "0".

Input R1 takes priority over input S. When the signal state is "1" on both inputs S and R1, the signal state of the specified operand is reset to "0".

The instruction is not executed if the signal state at the two inputs S and R1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

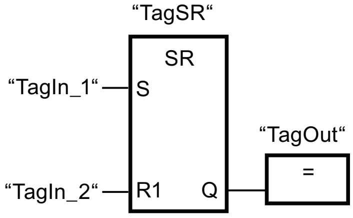
Parameters

The following table shows the parameters of the "Set/reset flip-flop" instruction:

Parameters	Declaration	Data type	Memory area	Description
S	Input	BOOL	I, Q, M, D, L	Enable setting
R1	Input	BOOL	I, Q, M, D, L	Enable resetting
<Operand>	Output	BOOL	I, Q, M, D, L	Operand that is set or reset
Q	Output	BOOL	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the "Set/reset flip-flop" instruction works:



The operands "TagSR" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagSR" and "TagOut" are reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 741)

RS: Reset/set flip-flop

Description

The "Set/reset flip-flop" instruction is used to reset or set the bit of a specified operand based on the signal state of the inputs R and S1. If the signal state is "1" at input R and "0" at input S1, the specified operand will be reset to "0". If the signal state is "0" at input R and "1" at input S1, the specified operand is set to "1".

Input S1 takes priority over input R. When the signal state is "1" at both inputs R and S1, the signal state of the specified operand is set to "1".

The instruction is not executed if the signal state at the two inputs R and S1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

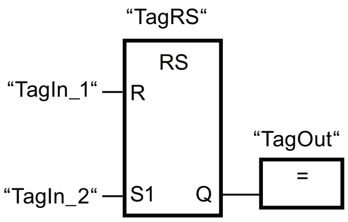
Parameters

The following table shows the parameters of the "Reset/set flip-flop" instruction:

Parameters	Declaration	Data type	Memory area	Description
R	Input	BOOL	I, Q, M, D, L	Enable resetting
S1	Input	BOOL	I, Q, M, D, L	Enable setting
<Operand>	Output	BOOL	I, Q, M, D, L	Operand that is reset or set.
Q	Output	BOOL	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the "Reset/set flip-flop" instruction works:



The operands "TagRS" and "TagOut" are reset when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagRS" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 741)

P: Scan operand for positive signal edge

Description

The "Scan operand for positive signal edge" instruction is used to determine whether there is a "0" to "1" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of the operand with the signal state of the previous query saved in an edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation from "0" to "1", there is a positive, rising edge.

If a falling edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit would be overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

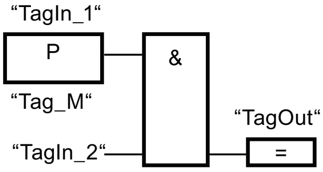
Parameters

The following table shows the parameters of the "Scan operand for positive signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	BOOL	I, Q, M, D, L	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit in which the signal state of the previous scan is saved

Example

The following example shows how the "Scan operand for positive signal edge" instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- There is a rising edge at input "TagIn_1".
- The signal state of the operand "TagIn_2" is "1".

See also

Overview of the valid data types (Page 741)

Example of detecting the direction of a conveyor belt (Page 1212)

N: Scan operand for negative signal edge

Description

The "Scan operand for negative signal edge" instruction is used to determine whether there is a "1" to "0" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of the operand with the signal state of the previous query saved in an edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation from "1" to "0", there is a negative, falling edge.

If a falling edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit would be overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

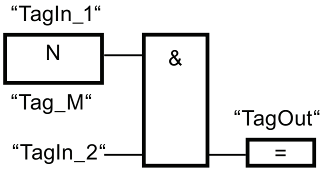
Parameters

The following table shows the parameters of the "Scan operand for negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	BOOL	I, Q, M, D, L	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit in which the signal state of the previous scan is saved

Example

The following example shows how the "Scan operand for negative signal edge" instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- There is a falling edge at input "TagIn_1".
- The signal state of the operand "TagIn_2" is "1".

See also

Overview of the valid data types (Page 741)

P=: Set operand on positive signal edge

Description

The "Set operand on positive signal edge" instruction is used to set a specified operand (<Operand2>) when there is a "0" to "1" change in the result of logic operation (RLO). The instruction compares the current result of logic operation with the result of logic operation from the previous query, which is saved in the edge memory bit (<Operand1>). If the instruction detects a change in the RLO from "0" to "1", there is a positive, rising edge.

When a positive edge is detected, <Operand2> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

You specify the operand (<Operand2>) to be set in the operand placeholder above the instruction. You specify the edge memory bit (<Operand1>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit would be overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

The instruction does not influence the RLO. The RLO at the input of the box is transferred directly to the output of the box.

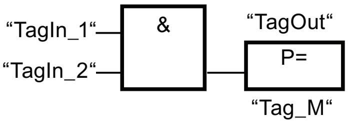
Parameters

The following table shows the parameters of the "Set operand on positive signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	InOut	BOOL	I, Q, M, D, L	Edge memory bit
<Operand2>	Output	BOOL	I, Q, M, D, L	Operand which is set when there is a positive signal edge.

Example

The following example shows the parameters of the "Set operand on positive signal edge" instruction:



The "TagOut" output is set for one program cycle, when the signal state at the input of the instruction box switches from "0" to "1" (positive signal edge). In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 741)

N=: Set operand on negative signal edge

Description

The "Set operand on negative signal edge" instruction is used to set a specified operand (<Operand1>) when there is a "1" to "0" change in the result of logic operation (RLO). The instruction compares the current RLO with the RLO from the previous query, which is saved in the edge memory bit (<Operand2>). If the instruction detects a change in the RLO from "1" to "0", there is a negative, falling edge.

When a negative edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

You specify the operand (<Operand1>) to be set in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit would be overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

The instruction does not influence the RLO. The RLO at the input of the box is transferred directly to the output of the box.

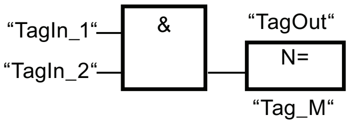
Parameters

The following table shows the parameters of the "Set operand on negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Output	BOOL	I, Q, M, D, L	Operand which is set when there is a negative signal edge.
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit

Example

The following example shows the mode of operation of the "Set operand on negative signal edge" instruction:



The operand "TagOut" is set for one program cycle if the signal state at the input of the instruction box changes from "1" to "0" (negative signal edge). In all other cases, the operand "TagOut" has the signal state "0".

See also

Overview of the valid data types (Page 741)

P_TRIG: Scan RLO for positive signal edge

Description

The "Scan RLO for positive signal edge" instruction is used to query a "0" to "1" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query, which is saved in an edge memory bit (<Operand>). If the instruction detects a change in the RLO from "0" to "1", there is a positive, rising edge.

If a rising edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit would be overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

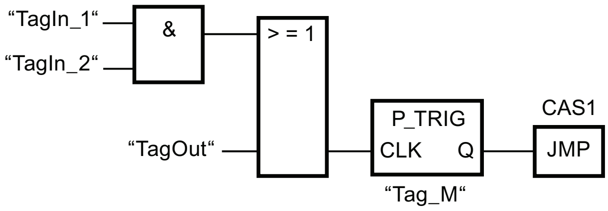
Parameters

The following table shows the parameters of the "Scan RLO for positive signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO.
<Operand>	InOut	BOOL	I, Q, M, D, L	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the preceding bit logic operation is saved in the edge memory bit "Tag_M". If a "0" to "1" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 741)

N_TRIG: Scan RLO for negative signal edge

Description

The "Scan RLO for negative signal edge" instruction is used to query a "1" to "0" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query saved in the edge memory bit (<Operand>). If the instruction detects a change in the RLO from "1" to "0", there is a negative, falling edge.

If a falling edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit would be overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

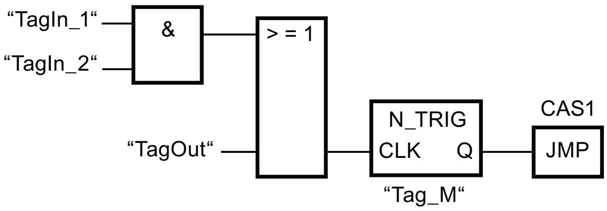
Parameters

The following table shows the parameters of the "Scan RLO for negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	I, Q, M, D, L	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the preceding bit logic operation is saved in the edge memory bit "Tag_M". If a "1" to "0" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 741)

Timer operations

TP: Generate pulse

Description

The "Generate pulse" instruction is used to set the Q output for the configured time duration PT. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive edge). The configured time duration PT begins when the instruction starts. Output Q is set for the time duration PT, regardless of the subsequent course of the input signal (postive edge). Even if a new rising signal edge is detected, the signal state at the output Q is not affected as long as the PT time duration is running.

The current time value can be queried at the ET output. The time value starts at T#0s and ends when the value of the time duration PT is reached. If the configuerd time duration PT is reached and the signal state at input IN is "0", the ET output is reset.

Each call of the "Generate pulse" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER or TP that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic refer to "See also".

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. The query of the data is only identical from the call of the instruciton to the next call of the instruction.

The execution of the "Generate pulse" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Generate pulse" instruction:

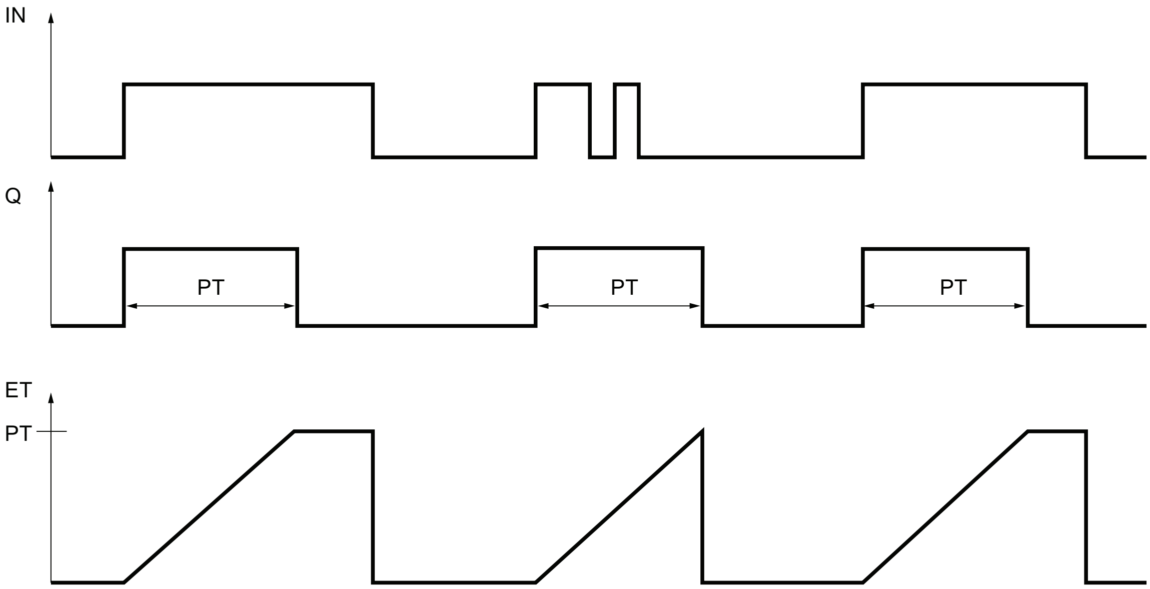
Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L or constant	Duration of the pulse. The value of the PT parameter must be positive.

Parameters	Declaration	Data type	Memory area	Description
Q	Output	BOOL	I, Q, M, D, L	Pulse output
ET	Output	TIME	I, Q, M, D, L	Current time value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate pulse" instruction:



See also

Overview of the valid data types (Page 741)

Example of controlling room temperature (Page 1217)

TON: Generate on-delay

Description

The "Generate on delay" instruction is used to delay the setting of the Q output for the configured time duration PT. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive edge). The programmed time duration PT begins when the instruction starts. When the time duration PT expires, the output Q has the signal state "1". Output Q remains set as long as the start input is still "1". When the signal state at the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new rising edge is detected at the start input.

The current time value can be queried at the ET output. The time value starts at T#0s and ends when the value of the time duration PT is reached. The ET output is reset as soon as the signal state at the IN input changes to "0".

Each call of the "Generate on-delay" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER or TON that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic refer to "See also".

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. The query of the data is only identical from the call of the instruciton to the next call of the instruction.

The execution of the "Generate on-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

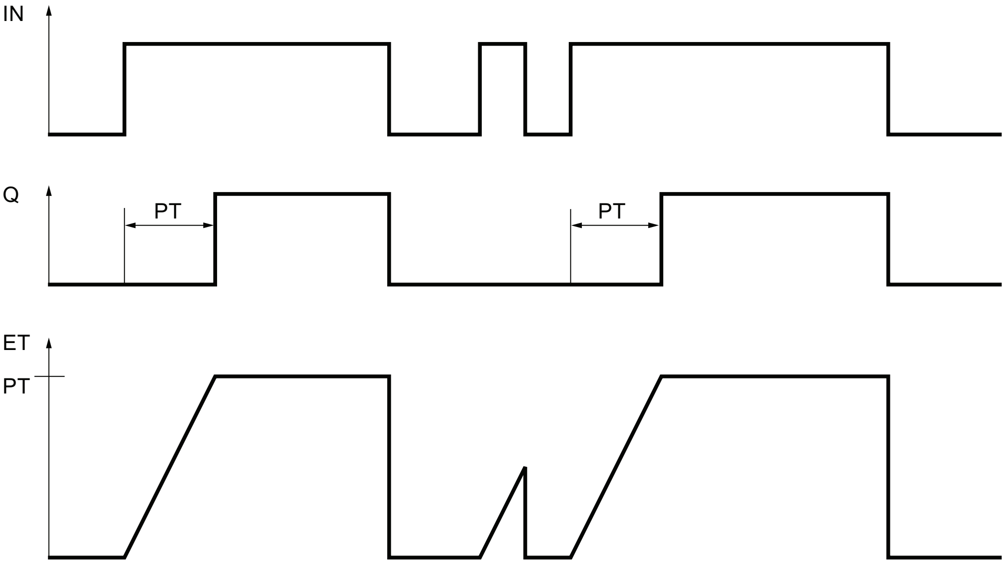
The following table shows the parameters of the "Generate on-delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L or constant	Duration of the on delay. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is set when the time PT expires.
ET	Output	TIME	I, Q, M, D, L	Current time value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate on-delay" instruction:



See also

Overview of the valid data types (Page 741)

TOF: Generate off-delay

Description

The "Generate off-delay" instruction is used to delay the resetting of the Q output for the configured time duration PT. The Q output is set when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). When the signal state at input IN changes back to "0" (positive signal edge), the configured time duration PT starts. Output Q remains set as long the time duration PT is running. When the time PT expires, the Q output is reset. If the signal state at the IN input changes to "1" before the time duration PT expires, the time is reset. The signal state at the output Q will continue to be "1".

The current time value can be queried at the ET output. The time value starts at T#0s and ends when the value of the time duration PT is reached. When the time PT expires, the ET output remains set to the current value until the IN input changes back to "1". If input IN switches to "1" before the time duration PT expires, the ET output is reset to the value T#0s.

Each call of the "Generate off-delay" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER or TOF that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic refer to "See also".

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. The query of the data is only identical from the call of the instruciton to the next call of the instruction.

The execution of the "Generate off-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

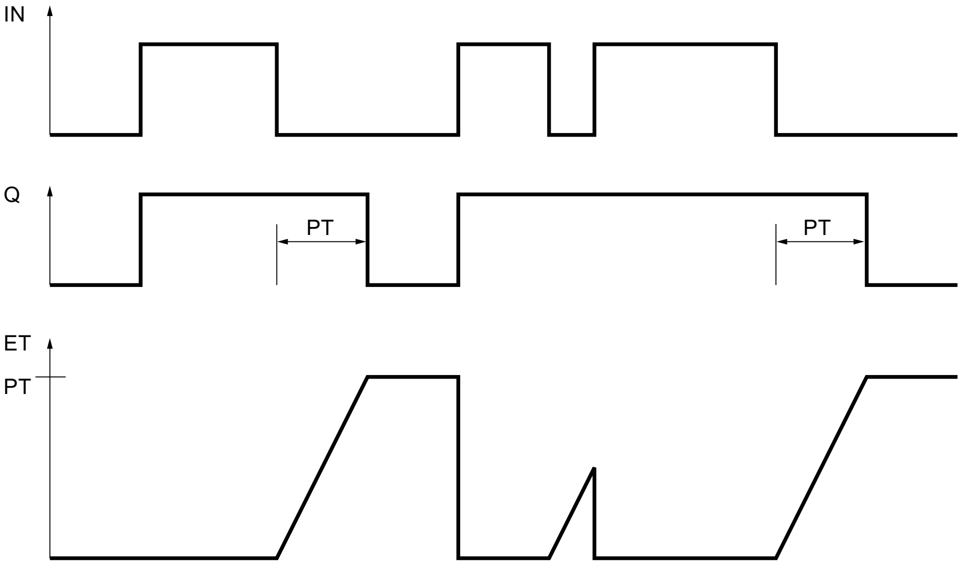
The following table shows the parameters of the "Generate off-delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L or constant	Duration of the off delay. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is reset when the timer PT expires.
ET	Output	TIME	I, Q, M, D, L	Current time value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate off-delay" instruction:



See also

Overview of the valid data types (Page 741)

TONR: Time accumulator

Description

The "Time accumulator" instruction accumulates time values within a period set by parameter PT. The instruction is executed and the configured time duration PT is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive edge). While the time set at PT is running, the time values are accumulated that are recorded at signal state "1" at input IN. The accumulated time is written to output ET and can be queried there. When the current time value PT is reached, the output Q has the signal state "1". Output Q remains set at "1", even when the signal state at input IN changes to "0".

The R input resets the outputs ET and Q regardless of the signal state at the start input.

Each call of the "Time accumulator" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER or TONR that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic refer to "See also".

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. The query of the data is only identical from the call of the instruciton to the next call of the instruction.

The execution of the "Time accumulator" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

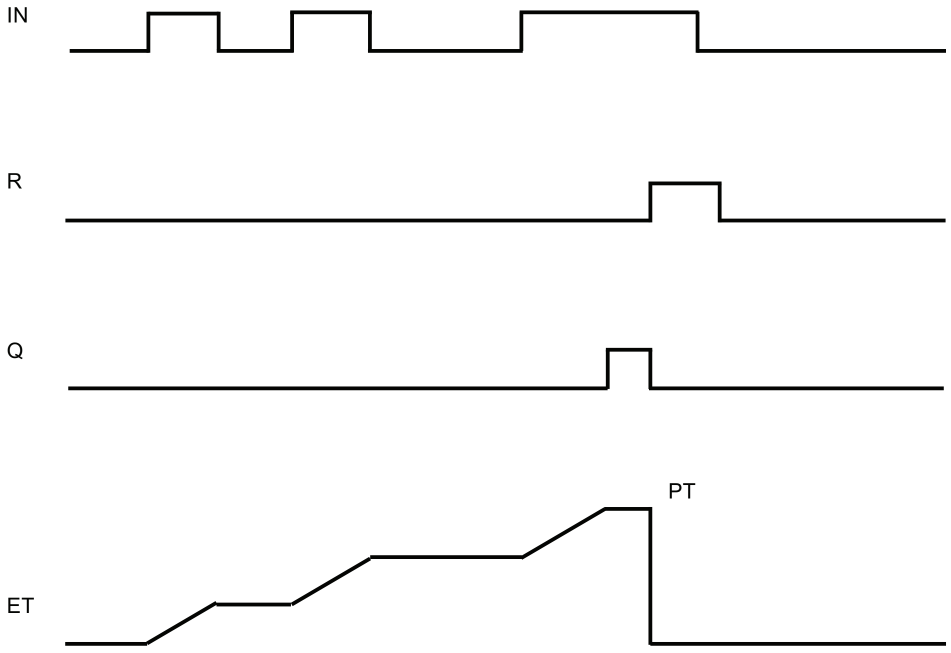
The following table shows the parameters of the "Time accumulator" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
R	Input	BOOL	I, Q, M, D, L	Reset input
PT	Input	TIME	I, Q, M, D, L or constant	Maximum duration of time recording. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is set when the time PT expires.
ET	Output	TIME	I, Q, M, D, L	Current time value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Time accumulator" instruction:



See also

Overview of the valid data types (Page 741)

TP: Start pulse timer

Description

The "Start pulse timer" instruction is used to start an IEC timer with the configured time duration as pulse. The IEC timer is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The IEC timer runs for the specified time duration regardless of the subsequent course of the the result of logic operation. The expiry of the IEC timer is also not affected by the detection of a new rising edge. As long as the IEC timer is running, the querying of the timer status for "1" returns the signal state "1". When the IEC timer has expired, the timer status returns the signal state "0".

The "Start pulse timer" instruction stores its data in a structure of the data type IEC_TIMER or TP. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. The query of the data is only identical from the call of the instruciton to the next call of the instruction.

The current timer status is saved in the structure components "Q" of the IEC timer. You can query the timer status with the help of a binary logic operation.

The execution of the "Start pulse timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

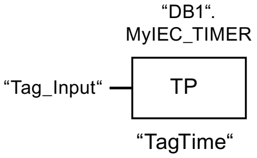
The following table shows the parameters of the instruction "Start pulse timer":

Parameters	Declaration	Data type	Memory area	Description
<Time duration>	Input	TIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs
<IEC timer>	InOut	IEC_TIMER/TP	D, L	IEC timer, which is started

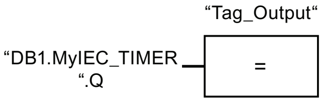
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Start pulse timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "0" to "1". Timer "DB1".MyIEC_TIMER starts running for the time duration that is stored in operand "TagTime".



As long as the timer "DB1".MyIEC_TIMER is running, the timer status ("DB1".MyIEC_TIMER.Q) has the signal state "1" and the operand "Tag_Output" is set. When the IEC timer has expired, the signal state of the timer status changes back to "0" and the "Tag_Output" operand is reset.

See also

Overview of the valid data types (Page 741)

TON: Start on-delay timer

Description

The "Start on-delay timer" instruction is used to start an IEC timer with the configured time duration as on-delay. The IEC timer is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The IEC timer runs for the specified time duration. The query of the timer status for "1" returns the signal state "1" if the timer ran out and the result of logic operation at the input of the instruction is "1". If the RLO changes to "0" before the end of the timer, the running IEC timer is reset. The query of the timer status for "1" returns the signal state "0". The IEC timer restarts when the next rising signal edge is detected at the input of the instruction.

The "Start on-delay timer" instruction stores its data in a structure of the data type IEC_TIMER or TON. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. The query of the data is only identical from the call of the instruciton to the next call of the instruction.

The current timer status is saved in the structure components "Q" of the IEC timer. You can query the timer status with the help of a binary logic operation.

The execution of the "Start on-delay timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

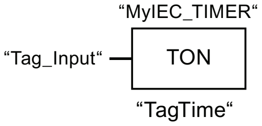
The following table shows the parameters of the instruction "Start on-delay timer":

Parameters	Declaration	Data type	Memory area	Description
<Time duration>	Input	TIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs
<IEC timer>	InOut	IEC_TIMER/TON	D, L	IEC timer, which is started

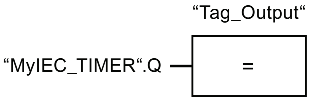
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Start on-delay timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "0" to "1". Timer "MyIEC_TIMER" starts running for the time duration that is stored in operand "TagTime".



If the timer "MyIEC_TIMER" has expired and the operand "Tag_Input" has the signal state "1", querying the timer status ("MyIEC_TIMER".Q) returns signal state "1" and the "Tag_Output" operand is set. When the signal state of the operand "Tag_Input" changes to "0", the querying of the timer status returns the signal state "0" and the operand "Tag_Output" is reset.

See also

Overview of the valid data types (Page 741)

TOF: Start off-delay timer

Description

The "Start off-delay timer" instruction is used to start an IEC timer with the configured time duration as off-delay. The query of the timer status for "1" returns the signal state "1" if the result of the logic operation (RLO) at the input of the instruction has the signal state "1". When the RLO changes from "1" to "0" (negative signal edge), the IEC timer starts with the specified time duration. The timer status remains at signal state "1" as long as the IEC timer is running. When the timer has run out and the RLO at the input of the instruction has the signal state "0", the timer status is set to the signal state "0". If the RLO changes to "1" before the end of the timer, the running IEC timer is reset and the timer status remain at the signal state "1".

The "Start off-delay timer" instruction stores its data in a structure of the data type IEC_TIMER or TOF. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. The query of the data is only identical from the call of the instruciton to the next call of the instruction.

The current timer status is saved in the structure components Q of the IEC timer. You can query the timer status with the help of a binary logic operation.

The execution of the "Start off-delay timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

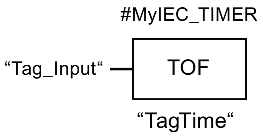
The following table shows the parameters of the instruction "Start off-delay timer":

Parameters	Declaration	Data type	Memory area	Description
<Time duration>	Input	TIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs
<IEC timer>	InOut	IEC_TIMER/TOF	D, L	IEC timer, which is started

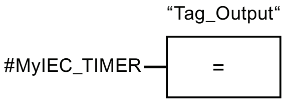
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Start off-delay timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "1" to "0". Timer "#MyIEC_TIMER" starts running for the time duration that is stored in operand "TagTime".



As long as the MyIEC_TIMER timer is running, the time status (#MyIEC_TIMER.Q) has signal state "1" and the "Tag_Output" operand is set. If the timer has expired and the operand "Tag_Input" has the signal state "0", the query of the timer status returns the signal state "0". If the signal state of the "Tag_Input" operand changes to "1" before the "#MyIEC_TIMER" timer expires, the time is reset. When the signal state of the operand "Tag_Input" is "1", the query of the timer status returns the signal state "1".

See also

Overview of the valid data types (Page 741)

TONR: Time accumulator

Description

The "Time accumulator" instruction is used to record how long the signal is at the input of instruction "1". The instruction is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The time is recorded as long at the RLO is "1". If the RLO changes to "0", the instruction is halted. If the RLO changes back to "1", the time recording is continued. The query of the timer status for "1" returns the signal state "1" if the recorded time exceeds the value of the specified time duration and the RLO at the input of coil is "1".

The timer status and the currently expired timer can be reset to "0" using the "Reset timer" instruction.

The "Time accumulator" instruction stores its data in a structure of the data type IEC_TIMER or TONR. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. The query of the data is only identical from the call of the instruciton to the next call of the instruction.

The current timer status is saved in the structure components Q of the IEC timer. You can query the timer status with the help of a binary logic operation.

The execution of the "Time accumulator" instruction requires a preceding logic operation. It can be placed only at the end of the network.

Parameters

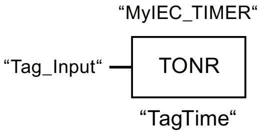
The following table shows the parameters of the "Time accumulator" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Time duration>	Input	TIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs
<IEC timer>	InOut	IEC_TIMER/TONR	D, L	IEC timer, which is started

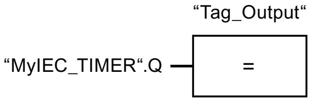
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Time accumulator" instruction is executed if there is a positive signal edge in the RLO. The time is recorded as long as the operand "Tag_Input" has the signal state "1".



If the recorded time exceeds the value of the operand "TagTime", then the query of the timer status ("MyIEC_TIMER".Q) will return the signal state "1" and the operand "Tag_Output" will be set.

See also

Overview of the valid data types (Page 741)

RT: Reset timer (Page 1430)

RT: Reset timer

Description

The instruction "Reset timer" is used to reset an IEC timer to "0". You specify the IEC timer to be reset by entering the name of the data block that contains the structure of the IEC timer in the placeholder above the instruction.

The instruction is only executed if the result of logic operation (RLO) at the box input is "1". When the instruction is executed the structure components of the IEC timer are reset to "0" in the specified data block. If the signal state at box input is "0", the instruction is not executed.

The instruction does not influence the RLO. The RLO at the box input is transferred directly to the box output.

You must assign a IEC timer declared in the program to the "Reset timer" instruction.

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. The query of the data is only identical from the call of the instruciton to the next call of the instruction.

Parameters

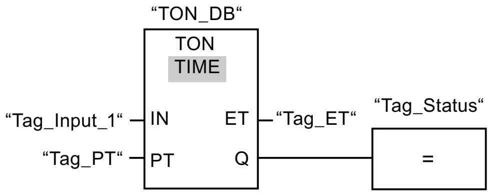
The following table shows the parameters of the instruction "Reset timer":

Parameters	Declaration	Data type	Memory area	Description
<IEC timer>	InOut	IEC_TIMER, TON, TOF, TP	D, L	IEC timer, which is reset.

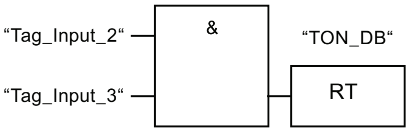
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Generate on-delay" instruction executes when the signal state of the "Tag_Input_1" operand changes from "0" to "1". The IEC timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PT".



If the operands "Tag_Input_2" and "Tag_Input_3" have the signal state "1", the "Reset timer" instruction is executed and the IEC timer stored in the data block "TON_DB" is reset.

See also

Overview of the valid data types (Page 741)

PT: Load time duration

Description

Use the "Load time duration" instruction to set the time duration of an IEC timer. The instruction is executed in every cycle when the result of logic operation at the input of the instruction has the signal state "1". The instruction writes the specified time duration to the structure of the specified IEC timer.

Note

If the specified IEC timer is running during the execution, the instruction overwrites the current time duration of the specified IEC timer. As a result, the timer status of the IEC timer can change.

You must assign a IEC timer declared in the program to the "Load time duration" instruction.

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. The query of the data is only identical from the call of the instruciton to the next call of the instruction.

Parameters

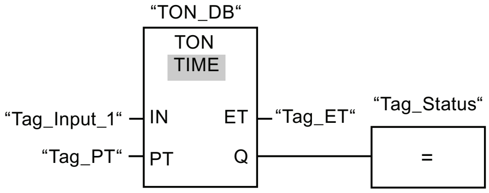
The following table shows the parameters of the instruction "Load time duration":

Parameters	Declaration	Data type	Memory area	Description
<Time duration>	Input	TIME	I, Q, M, D, L or constant	Time duration
<IEC timer>	InOut	IEC_TIMER, TON, TOF, TP	D, L	IEC timer, the duration of which is set.

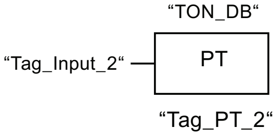
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Generate on-delay" instruction executes when the signal state of the "Tag_Input_1" operand changes from "0" to "1". The IEC timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PT".



The "Load time duration" instruction is executed when the operand "Tag_Input_2" has the signal state "1". The instruction writes the time duration "Tag_PT_2" in the instance data block "TON_DB" and at the same time overwrites the value of the operand "Tag_PT" within the data block. As a result, the signal state of the timer status can change at the next query.

Note

The "Tag_Input_2" is exected as pulse flag in order that the time duration is loaded only throughout one program cylce.

See also

Overview of the valid data types (Page 741)

Counter operations

CTU: Count up

Description

The "Count up" instruction is used to increment the value at the CV output. When the signal state at the CU input changes from "0" to "1" (positive signal edge), the instruction is executed and the current count value at the CV output is incremented by one. When the instruction executes for the first time, the current count value at the CV output is set to zero. The counter is incremented each time a positive signal edge is detected, until it reaches the high limit for the specified data type at output CV. When the high limit is reached, the signal state at the CU input no longer has an effect on the instruction.

You can scan the counter status at the Q output. The signal state at the Q output is determined by the PV parameter. If the current count value is greater than or equal to the value of the PV parameter, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0". You can also specify a constant for the PV parameter.

The value at the CV output is reset to "0" and saved to an edge memory bit when the signal state at input R changes to "1". As long as the R input has signal state "1", the signal state at the CU input has no effect on the instruction.

Each call of the "Count up" instruction must be assigned to an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

- Data block of system data type IEC_counter (global DB):
  - IEC_SCOUNTER / IEC_USCOUNTER
  - IEC_COUNTER / IEC_UCOUNTER
  - IEC_DCOUNTER / IEC_UDCOUNTER
- Local tag:
  - CTU_SINT / CTU_USINT
  - CTU_INT / CTU_UINT
  - CTU_DINT / CTU_UDINT

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_COUNTER (for example, "MyIEC_COUNTER")
- Declaration as a local tag of CTU type in the "InOut" or "Static" "Input" section of a block (for example, #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic refer to "See also".

The execution of the "Count up" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count up" instruction:

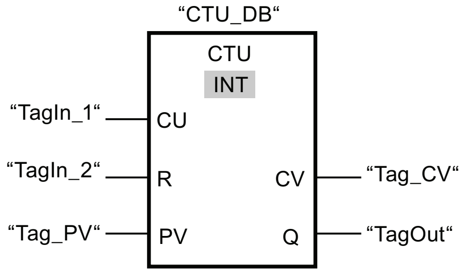
Parameters	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count input
R	Input	BOOL	I, Q, M, D, L	Reset input
PV	Input	Integers	I, Q, M, D, L or constant	Value at which the output Q is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	Integers	I, Q, M, D, L	Current count value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count up" instruction is executed and the current count value of the "Tag_CV" operand is incremented by one. With each further positive signal edge, the counter is incremented until the high limit value of the specified data type (INT = 32 767) is reached.

The value of the PV parameter is adopted as the limit for determining the "TagOut" output. The "TagOut" output has signal state "1" as long as the current count value is greater than or equal to the value of the "Tag_PV" operand. In all other cases, the "TagOut" ouput returns the signal state "0".

See also

Overview of the valid data types (Page 741)

CTD: Count down

Description

You can use the "Count down" instruction to decrement the value at output CV. When the signal state at the CD input changes from "0" to "1" (positive signal edge), the instruction is executed and the current count value at the CV output is decremented by one. When the instruction executes the first time, the count value of the CV parameter will be set to the value of the PV parameter. Each time a positive edge is detected, the count value is decremented until it reaches the low limit value of the specified data type. When the low limit is reached, the signal state at the CD input no longer has an effect on the instruction.

You can scan the counter status at the Q output. If the current count value is less than or equal to "0", the Q output is set to signal state "1". In all other cases, the Q output has signal state "0". You can also specify a constant for the PV parameter.

The value at the CV output is set to the value of the PV parameter and saved to a edge memory bit when the signal state at the LD input changes from "0" to "1". As long as the LD input has signal state "1", the signal state at the CD input has no effect on the instruction.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

- Data block of system data type IEC_counter (global DB):
  - IEC_SCOUNTER / IEC_USCOUNTER
  - IEC_COUNTER / IEC_UCOUNTER
  - IEC_DCOUNTER / IEC_UDCOUNTER
- Local tag:
  - CTU_SINT / CTU_USINT
  - CTU_INT / CTU_UINT
  - CTU_DINT / CTU_UDINT

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_COUNTER (for example, "MyIEC_COUNTER")
- Declaration as a local tag of CTD type in the "Input", "InOut" or "Static"section of a block (for example, #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The execution of the "Count down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count down" instruction:

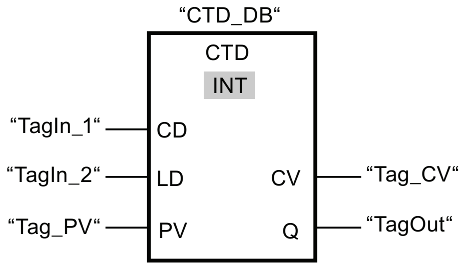
Parameters	Declaration	Data type	Memory area	Description
CD	Input	BOOL	I, Q, M, D, L	Count input
LD	Input	BOOL	I, Q, M, D, L	Load input
PV	Input	Integers	I, Q, M, D, L or constant	Value at which the output Q is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	Integers	I, Q, M, D, L	Current count value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count down" instruction executes and the value at the "Tag_CV" output is decremented by one. With each additional positive signal edge, the count value is decremented until the low limit of the specified data type (INT = -32 768) is reached.

The value of the PV parameter is adopted as the limit for determining the "TagOut" output. The "TagOut" output has signal state "1" as long as the current count value is less than or equal to "0". In all other cases, the "TagOut" output returns the signal state "0".

See also

Overview of the valid data types (Page 741)

CTUD: Count up and down

Description

You can use the "Count up and down" instruction to increment and decrement the count value at the CV output. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current count value is incremented by one and stored at the CV output. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the current count value at the CV output is decremented by one. If there is a positive signal edge at the CU and CD inputs in one program cycle, the current count value at the CV output remains unchanged.

The counter can be incremented until it reaches the high limit value of the data type specified at output CV. When the high limit value is reached, the count value is no longer incremented on a positive signal edge. When the low limit value of the specified data type is reached, the counter is not decremented any further.

When the signal state at the LD input changes to "1", the count value at the CV output is set to the value of the PV parameter and stored in a edge memory bit. As long as the LD input has signal state "1", the signal state at the CU and CD inputs has no effect on the instruction.

The count value is set to "0" and stored in an edge memory bit when the signal state at input R changes to "1". As long as the R input has signal state "1", a change in the the signal state of the CU, CD and LD inputs has no effect on the "Count up and down" instruction.

You can scan the current status of the up counter at the QU output. If the current count value is greater than or equal to the value of the PV parameter, the QU output is set to signal state "1". In all other cases, the QU output has signal state "0". You can also specify a constant for the PV parameter.

You can scan the current status of the down counter at the QD output. If the current count value is less than or equal to zero, the QD output is set to signal state "1". In all other cases, the QD output has signal state "0".

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

- Data block of system data type IEC_counter (global DB):
  - IEC_SCOUNTER / IEC_USCOUNTER
  - IEC_COUNTER / IEC_UCOUNTER
  - IEC_DCOUNTER / IEC_UDCOUNTER
- Local tag:
  - CTU_SINT / CTU_USINT
  - CTU_INT / CTU_UINT
  - CTU_DINT / CTU_UDINT

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_COUNTER (for example, "MyIEC_COUNTER")
- Declaration as a local tag of CTD type in the "Input", "InOut" or "Static"section of a block (for example, #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The execution of the "Count up and down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count up and down" instruction:

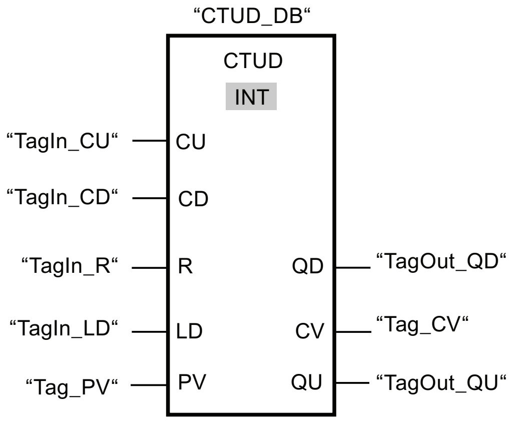
Parameters	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L	Count down input
R	Input	BOOL	I, Q, M, D, L	Reset input
LD	Input	BOOL	I, Q, M, D, L	Load input
PV	Input	Integers	I, Q, M, D, L or constant	Value at which the output QU / QD is set.
QU	Output	BOOL	I, Q, M, D, L	Up-counter status
QD	Output	BOOL	I, Q, M, D, L	Down-counter status
CV	Output	Integers	I, Q, M, D, L	Current count value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the signal state at the "TagIn_CU" or "TagIn_CD" input changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When there is a positive signal edge at the "TagIn_CU" input, the current count value is incremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the "TagIn_CD" input, the count value is decremented by one and stored at the "Tag_CV" output. When there is a positive edge at the CU input, the count value is incremented until it reaches the high limit value (INT = 32 767). If input CD has a positive signal edge, the count value is incremented until it reaches the low limit value (INT = -32 768).

The "TagOut_GU" output has signal state "1" as long as the current count value is greater than or equal to the value at the "Tag_PV" input. In all other cases, the "TagOut_QU" output returns the signal state "0".

The "TagOut_QD" output has signal state "1" as long as the current count value is less than or equal to "0". In all other cases, the "TagOut_QD" output has signal state "0".

See also

Overview of the valid data types (Page 741)

Example of detecting the fill level of a storage area (Page 1214)

Comparator operations

CMP ==: Equal

Description

The "Equal" instruction is used to query whether the value at input IN1 is equal to the value at input IN2.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their ASCII code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison.

The following table shows examples of string comparisons:

IN1	IN2	RLO of the instruction
'AA'	'AA'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0

The "Equal" instruction also compares individual characters of a string (STRING). The number of the character to be compared is specified in square brackets beside the operand name. With the specification "MyString[2]", for example, the second character of the "MyString" string is compared.

If IEC check is enabled, the operands to be compared must be of the same data type. If IEC check is not enabled, the width (length) of the operands must be the same. If the floating-point numbers are being compared, the operands to be compared must be of the same data type regardless of the IEC check setting.

Parameters

The following table shows the parameters of the instruction "Equal":

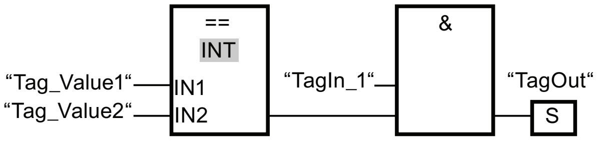
Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Bit strings, integers, floating-point numbers, characters, TIME, DATE, TOD, DTL	I, Q, M, D, L or constant	First value to compare
IN2	Input	Bit strings, integers, floating-point numbers, characters, TIME, DATE, TOD, DTL	I, Q, M, D, L or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" = "Tag_Value2").

See also

Overview of the valid data types (Page 741)

CMP <>: Not equal

Description

The "Not equal" instruction is used to query whether the value at input IN1 is not equal to the value at input IN2.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their ASCII code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison.

The following table shows examples of string comparisons:

IN1	IN2	RLO of the instruction
'AA'	'aa'	1
'Hello World'	'HelloWorld'	1
'AA'	'AA'	0

The "Not equal" instruction also compares individual characters of a string (STRING). The number of the character to be compared is specified in square brackets beside the operand name. With the specification "MyString[2]", for example, the second character of the "MyString" string is compared.

If IEC check is enabled, the operands to be compared must be of the same data type. If IEC check is not enabled, the width (length) of the operands must be the same. If the floating-point numbers are being compared, the operands to be compared must be of the same data type regardless of the IEC check setting.

Parameters

The following table shows the parameters of the instruction "Not equal":

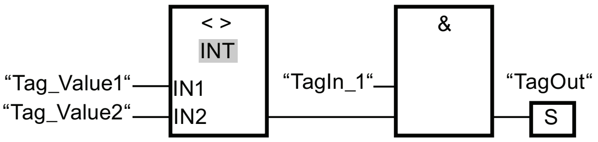
Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Bit strings, integers, floating-point numbers, characters, TIME, DATE, TOD, DTL	I, Q, M, D, L or constant	First value to compare
IN2	Input	Bit strings, integers, floating-point numbers, characters, TIME, DATE, TOD, DTL	I, Q, M, D, L or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" <> "Tag_Value2").

See also

Overview of the valid data types (Page 741)

CMP >=: Greater or equal

Description

The "Greater or equal" instruction is used to query whether the value at input IN1 is greater than or equal to the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their ASCII code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered greater.

The following table shows examples of string comparisons:

IN1	IN2	RLO of the instruction
'BB'	'AA'	1
'AAA'	'AA'	1
'Hello World'	'Hello World'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0
'AAA'	'a'	0

The "Greater or equal" instruction also compares individual characters of a string (STRING). The number of the character to be compared is specified in square brackets beside the operand name. With the specification "MyString[2]", for example, the second character of the "MyString" string is compared.

In comparing timer values, the RLO of the instruction is "1" if the timer at input IN1 is greater (more recent) than or equal to the timer at input IN2.

Parameters

The following table shows the parameters of the instruction "Greater or equal":

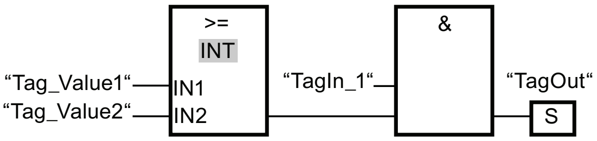
Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers, characters, TIME, TOD, DTL	I, Q, M, D, L or constant	First value to compare
IN2	Input	Integers, floating-point numbers, characters, TIME, TOD, DTL	I, Q, M, D, L or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" >= "Tag_Value2").

See also

Overview of the valid data types (Page 741)

Example of detecting the fill level of a storage area (Page 1214)

CMP <=: Less or equal

Description

The "Less or equal" instruction is used to query whether the value at input IN1 is less than or equal to the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their ASCII code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string is considered smaller.

The following table shows examples of string comparisons:

IN1	IN2	RLO of the instruction
'AA'	'aa'	1
'AAA'	'a'	1
'Hello World'	'Hello World'	1
'HelloWorld'	'Hello World'	0
'BB'	'AA'	0
'AAA'	'AA'	0

The "Less or equal" instruction also compares individual characters of a string (STRING). The number of the character to be compared is specified in square brackets beside the operand name. With the specification "MyString[2]", for example, the second character of the "MyString" string is compared.

In comparing timer values, the RLO of the instruction is "1" if the timer at input IN1 is less (less recent) than or equal to the time at input IN2.

Parameters

The following table shows the parameters of the instruction "Less or equal":

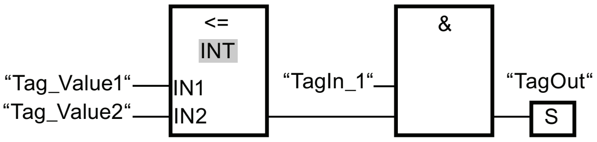
Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers, characters, TIME, TOD, DTL	I, Q, M, D, L or constant	First value to compare
IN2	Input	Integers, floating-point numbers, characters, TIME, TOD, DTL	I, Q, M, D, L or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" <= "Tag_Value2").

See also

Overview of the valid data types (Page 741)

CMP >: Greater than

Description

The "Greater than" instruction is used to query whether the value at input IN1 is greater than the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their ASCII code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered greater.

The following table shows examples of string comparisons:

IN1	IN2	RLO of the instruction
'BB'	'AA'	1
'AAA'	'AA'	1
'AA'	'aa'	0
'AAA'	'a'	0

The "Greater than" instruction also compares individual characters of a string (STRING). The number of the character to be compared is specified in square brackets beside the operand name. With the specification "MyString[2]", for example, the second character of the "MyString" string is compared.

In comparing timer values, the RLO of the instruction is "1" if the timer at input IN1 is greater (more recent) than the timer at input IN2.

Parameters

The following table shows the parameters of the instruction "Greater than":

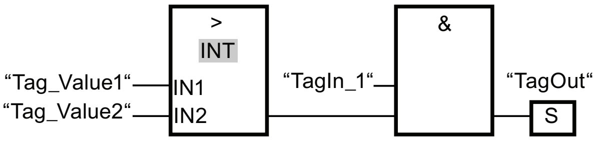
Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers, characters, TIME, TOD, DTL	I, Q, M, D, L or constant	First value to compare
IN2	Input	Integers, floating-point numbers, characters, TIME, TOD, DTL	I, Q, M, D, L or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" > "Tag_Value2").

See also

Overview of the valid data types (Page 741)

CMP <: Less than

Description

The "Less than" instruction is used to query whether the value at input IN1 is less than the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their ASCII code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string is considered smaller.

The following table shows examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'aa'	1
'AAA'	'a'	1
'BB'	'AA'	0
'AAA'	'AA'	0

The "Less than" instruction also compares individual characters of a string (STRING). The number of the character to be compared is specified in square brackets beside the operand name. With the specification "MyString[2]", for example, the second character of the "MyString" string is compared.

In comparing timer values, the RLO of the instruction is "1" if the timer at input IN1 is less (less recent) than the timer at input IN2 .

Parameters

The following table shows the parameters of the instruction "Less than":

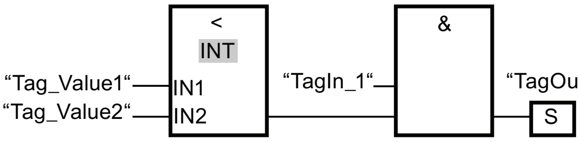
Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers, characters, TIME, TOD, DTL	I, Q, M, D, L or constant	First value to compare
IN2	Input	Integers, floating-point numbers, characters, TIME, TOD, DTL	I, Q, M, D, L or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled (("Tag_Value1" < "Tag_Value2")).

See also

Overview of the valid data types (Page 741)

Example of detecting the fill level of a storage area (Page 1214)

IN_RANGE: Value within range

Description

You can use the "Value within range" instruction to query whether of the value at input VAL is within a specific value range.

You specify the limits of the value range with the MIN and MAX inputs. The "Value within range" instruction compares the value at input VAL with the values of the inputs MIN and MAX and sends the result to the box output. If the value at input VAL satisfies the comparison MIN <= VAL or VAL <= MAX, the box output has the signal state "1". If the comparison is not fulfilled, the signal state is "0" at the box output.

The comparison function can only execute if the values to be compared are of the same data type. You can also specify constants at the MIN, MAX and VAL inputs.

Parameters

The following table shows the parameters of the "Value within range" instruction:

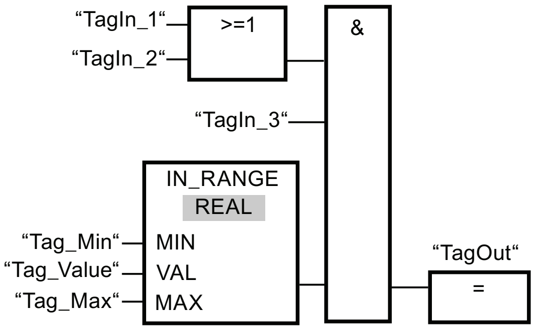
Parameters	Declaration	Data type	Memory area	Description
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VAL	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Comparison value
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
Box output	Output	BOOL	I, Q, M, D, L	Result of the comparison

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "1".
- The value of the operand "Tag_Value" is within the value range that is specified by the current values of the operands "Tag_Min" and "Tag_Max" (MIN <= VAL or VAL <= MAX).

See also

Overview of the valid data types (Page 741)

OUT_RANGE: Value outside range

Description

You can use the "Value outside range" instruction to query whether the value at input VAL is outside a specific range.

You specify the limits of the value range with the MIN and MAX inputs. The "Value outside range" instruction compares the value at input VAL with the values of the inputs MIN and MAX and sends the result to the box output. If the value at input VAL satisfies the comparison MIN > VAL or VAL > MAX, the box output has the signal state "1". The box output has the signal state "1" if a specified operand of data type REAL has an invalid value.

The box output returns the signal state "0", if the value at input VAL does not satisfy the MIN > VAL or VAL > MAX condition.

The comparison function can only execute if the values to be compared are of the same data type. You can also specify constants at the MIN, MAX and VAL inputs.

Parameters

The following table shows the parameters of the "Value outside range" instruction:

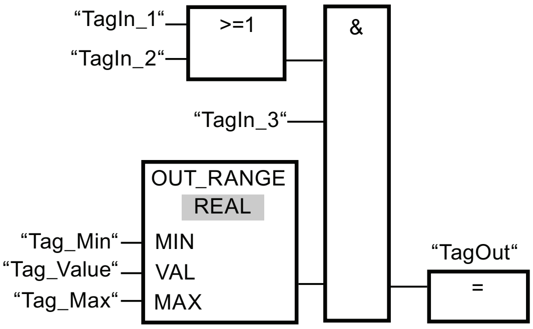
Parameters	Declaration	Data type	Memory area	Description
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VAL	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Comparison value
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
Box output	Output	BOOL	I, Q, M, D, L	Result of the comparison

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "1".
- The value of the operand "Tag_Value" is outside the value range that is specified by the values of the operands "Tag_Min" and "Tag_Max" (MIN > VAL or VAL > MAX).

See also

Overview of the valid data types (Page 741)

OK: Check validity

Description

You can use the "Check validity" instruction to check if the value of an operand (<operand>) is a valid floating-point number. The check is performed in every program cycle. If the operand value at the time of the query is a valid floating-point number, the output box will return the signal state"1". In all other cases, the signal state at the output of the "Check validity" instruction is "0".

You can use the "Check validity" instruction together with the EN mechanism. If you connect the instruction box to an enable input EN, the enable input is set only when the result of the validity query of the value is positive. You can use this function to ensure that an instruction is enabled only when the value of the specified operand is a valid floating-point number.

Parameters

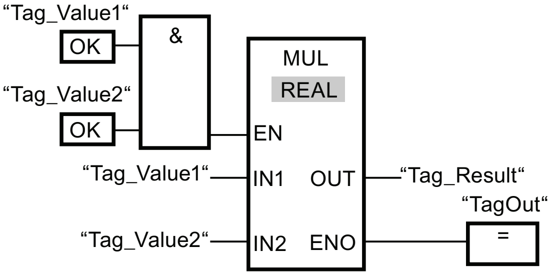
The following example shows how the "Check validity" instruction works:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L or constant	Value to be checked.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the "Check validity" instruction works:



When the values of the operands "Tag_Value1" and "Tag_Value2" show valid floating-point numbers, the "Multiply" (MUL) instruction is activated and the ENO output is set. During the execution of the "Multiply" (MUL) instruction, the value of the operand "Tag_Value1" is multiplied by the value of operand "Tag_Value2". The product of the multiplication is then stored in the operand "Tag_Result". If no errors occur during the execution of the instruction, the outputs ENO and "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 741)

NOT_OK: Check invalidity

Description

You can use the "Check invalidity" instruction to check if the value of an operand (<operand>) is an invalid floating-point number. The check is performed in every program cycle. If the operand value at the time of the query is a valid floating-point number, then the output box will return the signal state "1". In all other cases, the signal state on the output box is "0".

Parameters

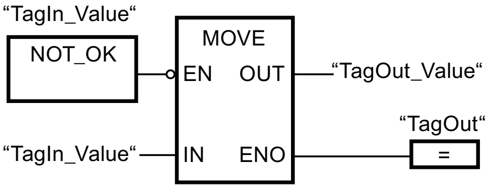
The following table shows the parameters of the instruction "Check invalidity":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L or constant	Value to be checked.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



When the value of operand "TagIn_Value" is an invalid floating-point number, the "Copy value" (MOVE) instruction will not be executed. The "TagOut" output is reset to signal state "0".

See also

Overview of the valid data types (Page 741)

Math functions

CALCULATE: Calculate

Description

The "Calculate" instruction is used to define and execute an expression (formula) for the calculation of mathematical operations or complex logic operations depending on the selected data type.

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box. Depending on the selected data type, you can combine the functionality of specific instructions to execute a complex calculation. The expression to be calculated is specified via a dialog you can open via the "Calculator" icon at the top right edge of the instruction box. The expression can contain the names of the input parameters and the syntax of the instructions. It is not permitted to specify operand names or operand addresses.

The following table shows the instructions that, depending on the selected data type, can be combined and executed in the expression of the "Calculate" instruction:

Data type	Instruction	Syntax	Example
Bit strings	AND: AND logic operation	AND	IN1 AND IN2 OR IN3
	OR: OR logic operation	OR	
	XOR: EXCLUSIVE OR logic operation	XOR	
	INV: Create ones complement	NOT	
	SWAP: Swap	SWAP	
Integers	ADD: Add	+	(IN1 + IN2) * IN3; (ABS(IN2))*(ABS(IN1))
	SUB: Subtract	-	
	MUL: Multiply	*	
	DIV: Divide	/	
	MOD: Return remainder of division	MOD	
	INV: Create ones complement	NOT	
	NEG: Create twos complement	-(in1)	
	ABS: Form absolute value	ABS( )	
Floating-point numbers	ADD: Add	+	((SIN(IN2)*SIN(IN2)+(SIN(IN3)*SIN(IN3))/IN3; (SQR(SIN(IN2))+(SQR(COS(IN3))/IN2
	SUB: Subtract	-	
	MUL: Multiply	*	
	DIV: Divide	/	
	NEG: Create twos complement	**	
	ABS: Form absolute value	ABS( )	
	SQR : Form square	SQR( )	
	SQRT: Form square root	SQRT( )	
	LN : Form natural logarithm	LN( )	
	EXP : Form exponential value	EXP( )	
	FRAC: Return fraction	FRAC( )	

Data type	Instruction	Syntax	Example
	SIN: Form sine value	SIN( )	
	COS: Form cosine value	COS( )	
	TAN: Form tangent value	TAN( )	
	ASIN: Form arcsine value	ASIN( )	
	ACOS: Form arccosine value	ACOS( )	
	ATAN: Form arctangent value	ATAN( )	
	NEG: Create twos complement	-(in1)	
	TRUNC: Truncate numerical value	TRUNC( )	
	ROUND: Round numerical value	ROUND( )	
	CEIL: Generate next higher integer from floating-point number	CEIL( )	
	FLOOR: Generate next lower integer from floating-point number	FLOOR( )	

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box.

The values of the input parameters are use to execute the specified expression. Not all defined input parameters have to be used in the expression. The result of the instruction is transferred to the box output OUT.

If, in the expression, you use inputs that are not available in the box, these inputs are automatically inserted. Provided that there are no gaps in the numbering of the inputs that are to be newly defined in the expression. You cannot, for example, use the input IN4 in the expression if the input IN3 is not defined.

The "Calculate" instruction is only executed if the signal state at the enable input EN is "1". If all individual instructions of the specified expression are executed error-free, the enable output ENO also has the signal state "1"

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- The result of the "Calculate" instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.
- An error occurred during the execution of one of the instructions specified in the expression.

Parameters

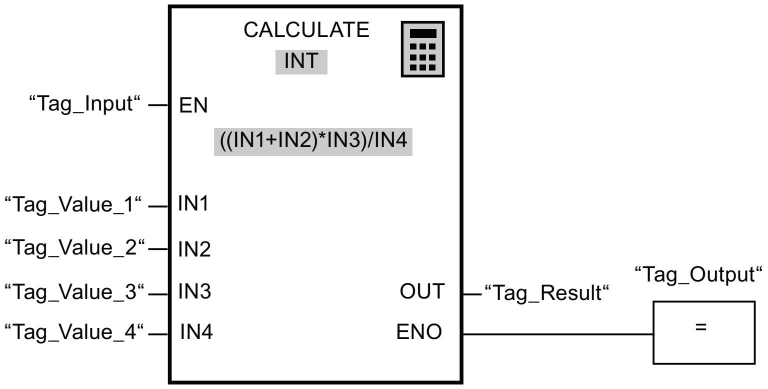
The following table shows the parameters of the instruction "Calculate":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L or constant	First available input
IN2	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L or constant	Second available input
INn	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L or constant	Additionally inserted inputs
OUT	Output	Bit strings, integers, floating-point numbers	I, Q, M, D, L	Output to which the end result is to be transferred.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	"Tag_Value_1"	4
IN2	"Tag_Value_2"	4
IN3	"Tag_Value_3"	3
IN4	"Tag_Value_4"	2
OUT	"Tag_Result"	12

The "Calculate" instruction is executed when input "Tag_Input" has the signal state "1". The value of operand "Tag_Value_1" is added to the value of operand "Tag_Value_2". The sum is multiplied with the value of the operand "Tag_Value_3". The product is divided by the value of the operand "Tag_Value_4". The quotient is transferred as end result to the operand "Tag_Result" at the OUT output of the instruction. If no errors occur during the execution of the individual instructions, output ENO and the operand "Tag_Output" are set to "1".

See also

Overview of the valid data types (Page 741)

Adding additional inputs and outputs to FBD elements (Page 965)

Removing instruction inputs and outputs (Page 966)

Basics of the EN/ENO mechanism (Page 819)

ADD: Add

Description

You can use the "Add" instruction to add the value at input IN1 and the value at input IN2 and query the sum at output OUT(OUT = IN1+IN2).

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box. During the execution of the instruction, the values of all available input parameters are added. The sum is stored at output OUT.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Add":

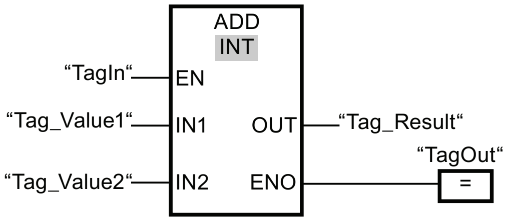
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	First number to be added
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Second number to be added
INn	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Optional input values, which are added.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Sum

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Add" instruction is executed. The value of operand "Tag_Value1" is added to the value of operand "Tag_Value2". The result of the addition is stored in the operand "Tag_Result". If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Removing instruction inputs and outputs (Page 966)

Basics of the EN/ENO mechanism (Page 819)

Adding additional inputs and outputs to FBD elements (Page 965)

SUB: Subtract

Description

You can use the "Subtract" instruction to subtract the value at input IN2 and the value at input IN1 and query the difference at output OUT (OUT = IN1-IN2).

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Subtract":

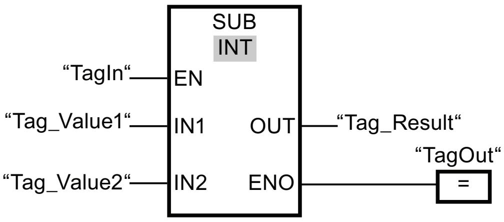
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Minuend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Subtrahend
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Difference

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Subtract" instruction is executed. The value of operand "Tag_Value2" is subtracted from the value of operand "Tag_Value1". The result of the subtraction is stored in the operand "Tag_Result". If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

MUL: Multiply

Description

The "Multiply" instruction is used to multiply the value at input IN1 with the value at input IN2 and query the product at output OUT(OUT = IN1*IN2).

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are multiplied. The product is stored at the OUT output.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- The result is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Multiply":

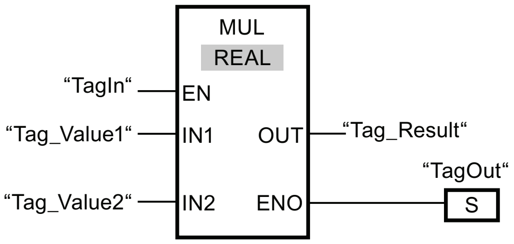
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	First value for multiplication
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Second value for multiplication
INn	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Optional input values, which are multiplied.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Product

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Multiply" instruction is executed. The value of operand "Tag_Value1" is multiplied with the value of operand "Tag_Value2". The result of the multiplication is stored in the operand "Tag_Result". If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Removing instruction inputs and outputs (Page 966)

Basics of the EN/ENO mechanism (Page 819)

Adding additional inputs and outputs to FBD elements (Page 965)

DIV: Divide

Description

You can use the "Divide" instruction to divide the value at input IN1 by the value at the input IN2 and query the quotient at output OUT (OUT = IN1/IN2).

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Divide":

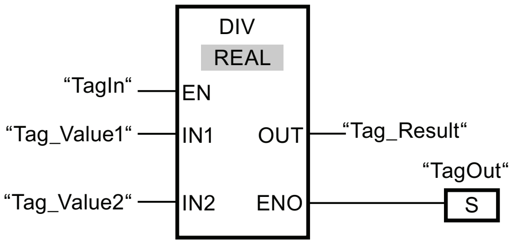
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Dividend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Divisor
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Quotient value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Divide" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The result of the division is stored in the operand "Tag_Result". If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

MOD: Return remainder of division

Description

The "Return remainder of division" instruction is used to divide the value at input IN1 by the value at input IN2 and query the remainder at output OUT.

The instruction is only executed if the signal state at input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

The instruction is not executed if the signal state at input EN is "0". In this case, output ENO is reset.

Parameters

The following table shows the parameters of the instruction "Return remainder of division":

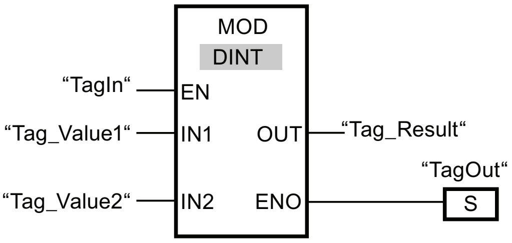
Parameters		Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers	I, Q, M, D, L or constant	Dividend
IN2	Input	Integers	I, Q, M, D, L or constant	Divisor
OUT	Output	Integers	I, Q, M, D, L	Remainder of division

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Return remainder of division" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The remainder of division is stored in operand "Tag_Result". If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

NEG: Create twos complement

Description

The "Create twos complement" instruction is used to change the sign of the value at input IN and query the result at output OUT. If there is a positive value at input IN, for example, the negative equivalent of this value is sent to output OUT.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Create twos complement" instruction:

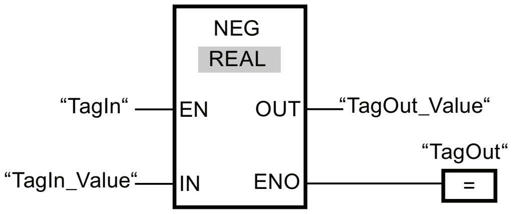
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	SINT, INT, DINT, REAL, LREAL	I, Q, M, D, L or constant	Input value
OUT	Output	SINT, INT, DINT, REAL, LREAL	I, Q, M, D, L	Twos complement of the input value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Create twos complement" instruction is executed. The sign of the value at input "TagIn_Value" is changed and the result is stored at output "TagOut_Value". If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

Adding additional inputs and outputs to FBD elements (Page 965)

INC: Increment

Description

You can use the "Increment" instruction to change the value of the operand at parameter IN/OUT to the next higher value and query the result. The "Increment" instruction is only started if the signal state at enable input EN is "1". If no overflow error occurs during execution, output ENO also has the signal state "1".

If the signal state at enable input EN is "0", the instruction is not executed. In this case, the enable output ENO is reset.

Parameters

The following table shows the parameters of the "Increment" instruction:

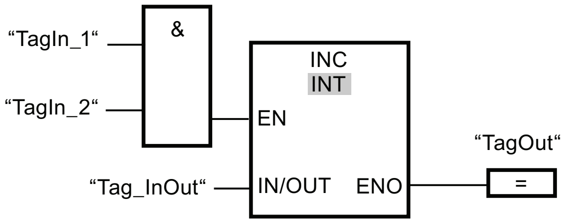
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN/OUT	InOut	Integers	I, Q, M, D, L	Value to be incremented.

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operands TagIn_1 and TagIn_2 have the signal state "1", the value of the operand "Tag_InOut" is incremented by one and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

Adding additional inputs and outputs to FBD elements (Page 965)

DEC: Decrement

Description

You can use the "Decrement" instruction to change the value of the operand at parameter IN/OUT to the next lower value and query the result. The "Decrement" instruction is only started if the signal state at enable input EN is "1". If the value range of the selected data type is not exceeded during the processing, the ENO output also has signal state "1".

If the signal state at enable input EN is "0", the instruction is not executed. In this case, the enable output ENO is reset.

Parameters

The following table shows the parameters of the instruction "Decrement":

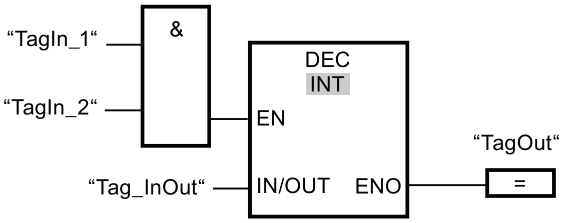
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN/OUT	InOut	Integers	I, Q, M, D, L	Value to be decremented.

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operands TagIn_1 and TagIn_2 have the signal state "1", the value of the operand "Tag_InOut" is decremented by one and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

Adding additional inputs and outputs to FBD elements (Page 965)

ABS: Form absolute value

Description

The "Form absolute value" instruction is used to calculate the absolute amount of the value specified at input IN. The result of the instruction is provided at output OUT and can be queried there.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Form absolute value":

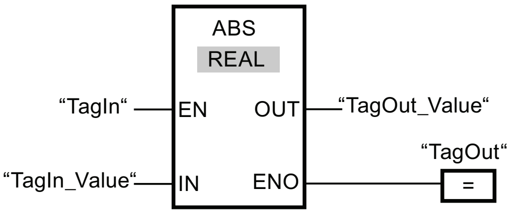
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	SINT, INT, DINT, floating-point numbers	I, Q, M, D, L or constant	Input value
OUT	Output	SINT, INT, DINT, floating-point numbers	I, Q, M, D, L	Absolute value of the input value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	"TagIn_Value"	- 6, 234
OUT	"TagOut_Value"	6, 234

If the operand "TagIn" has the signal state "1", the "Form absolute value" instruction is executed. The instruction calculates the absolute value of the value at input "TagIn_Value" and sends the result to output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

Adding additional inputs and outputs to FBD elements (Page 965)

MIN: Get minimum

Description

The "Get minimum" instruction compares the values at the available inputs and writes the lowest value to output OUT. In its initial state the instruction box contains at least 2 (IN1 and IN2) and not more than 100 inputs. The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box.

The instruction can only be executed if the tags on all inputs are of the same data type and the enable input EN has the signal state "1". If no errors occur during the execution of the instruction, enable output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable Input EN has the signal state "0".
- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get minimum" instruction:

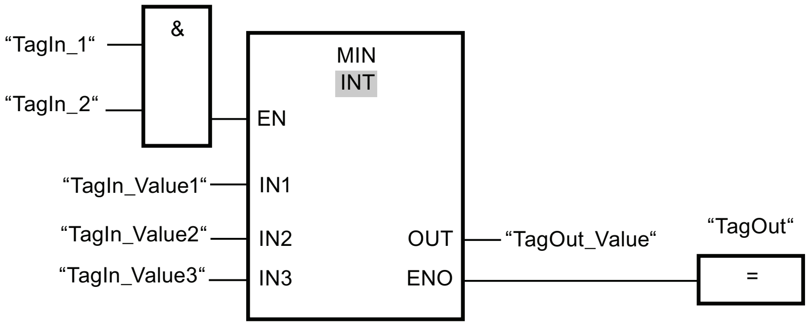
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Second input value
INn	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Additionally inserted inputs whose values are to be compared.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	TagIn_Value1	12 222
IN2	TagIn_Value2	14 444
IN3	TagIn_Value3	13 333
OUT	TagOut_Value	12 222

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Get minimum" instruction is executed. The instruction compares the values of the specified operands and copies the lowest value ("TagIn_Value1") to output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Removing instruction inputs and outputs (Page 966)

Basics of the EN/ENO mechanism (Page 819)

Adding additional inputs and outputs to FBD elements (Page 965)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values of the available inputs and writes the highest value to output OUT. In its initial state the instruction box contains at least 2 (IN1 and IN2) and not more than 100 inputs. The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box.

The instruction can only be executed if the tags on all inputs are of the same data type and the enable input EN has the signal state "1". If no errors occur during the execution of the instruction, enable output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable Input EN has the signal state "0".
- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get maximum" instruction:

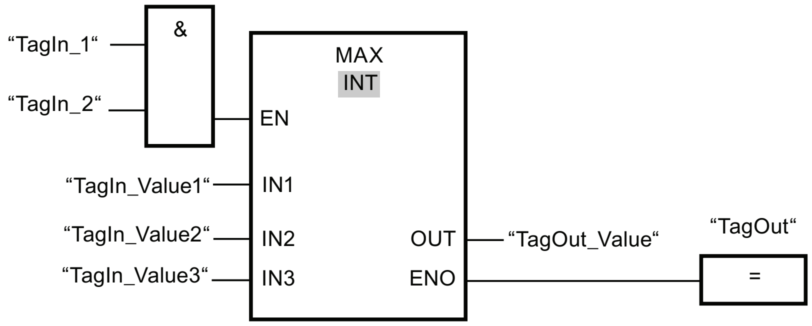
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	First input value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Second input value
INn	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Additionally inserted inputs whose values are to be compared.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	TagIn_Value1	12 222
IN2	TagIn_Value2	14 444
IN3	TagIn_Value3	13 333
OUT	TagOut_Value	14 444

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Get maximum" instruction is executed. The instruction compares the values of the specified operands and copies the greatest value ("TagIn_Value2") to output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Removing instruction inputs and outputs (Page 966)

Basics of the EN/ENO mechanism (Page 819)

Adding additional inputs and outputs to FBD elements (Page 965)

LIMIT: Set limit value

Description

The "Set limit value" instruction is used to limit the value at input IN to the values at the inputs MN and MX. If the value at the IN input meets the MN < IN < MX condition, it is copied to the OUT output. If the condition is not fulfilled and the input value IN is below the low limit MN, the output OUT is set to the value of the input MN. If the high limit MX is exceeded, output OUT is set to the value of input MX.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, enable output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable Input EN has the signal state "0".
- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value at input MN is greater than the value at input MX.

Parameters

The following table shows the parameters of the "Set limit value" instruction:

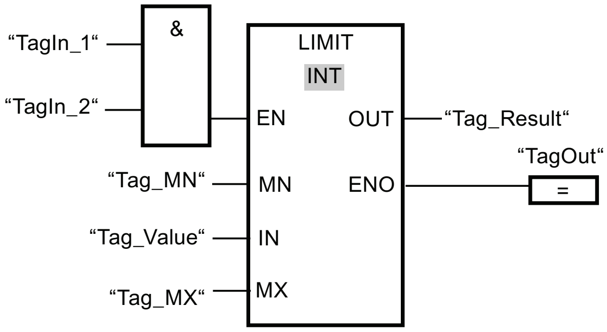
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit
IN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Input value
MX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
MN	Tag_MN	12 000
IN	Tag_Value	8 000
MX	Tag_MX	16 000
OUT	Tag_Result	12 000

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Set limit value" instruction is executed. The value of operand "Tag_Value" is compared with the values of operands "Tag_MN" and "Tag_MX". Since the value at the operand "Tag_Value" is less than the low limit, the value of the operand "Tag_MN" is copied to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

Adding additional inputs and outputs to FBD elements (Page 965)

SQR: Form square

Description

You can use the "Form square" instruction to square the value at input IN and query the result at output OUT.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form square":

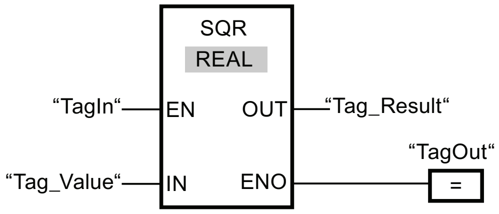
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L	Square of the input value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	5.0
OUT	Tag_Result	25.0

If the operand "TagIn" has the signal state "1", the "Form square" instruction is executed. The instruction squares the value of the operand "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

SQRT: Form square root

Description

You can use the "Form square root" instruction to calculate the square root of the value at input IN and query the result at output OUT. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number. If the value at input IN is "0", then the result is also "0".

The instruction is only executed if the signal state at the enable input is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is negative.

Parameters

The following table shows the parameters of the instruction "Form square root":

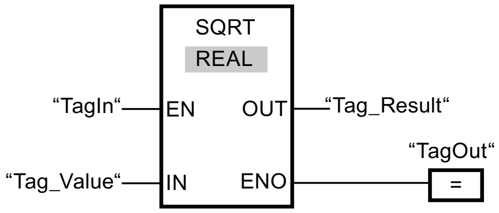
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L	Square root of the input value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	25.0
OUT	Tag_Result	5.0

If the operand "TagIn" has the signal state "1", the "Form square root" instruction is executed. The instruction calculates the square root of the operand "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

LN: Form natural logarithm

Description

The "Form natural logarithm" instruction is used to calculate the natural logarithm to base e (e = 2.718282e+00) from the value at input IN. The result is sent to output OUT and can be queried there. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, enable output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable Input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is negative.

Parameters

The following table shows the parameters of the instruction "Form natural logarithm":

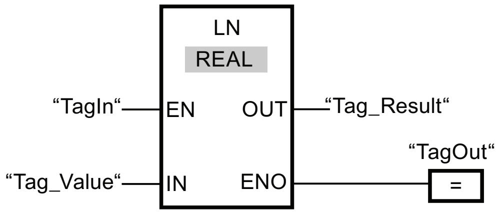
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L	Natural logarithm of the floating-point number

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Form natural logarithm" instruction is executed. The instruction forms the natural logarithm of the value at input "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

EXP: Form exponential value

Description

The instruction "Form exponential value" is used to calculate the power from the basis e (e = 2.718282e+00) and the value specified at input IN. The result is provided at the OUT output and can be queried there (OUT = e^{IN}).

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable Input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form exponential value":

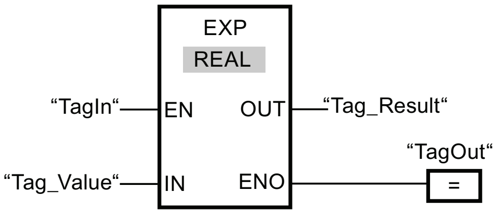
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L	Exponential value of the input value IN

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Form exponential value" instruction is executed. The instruction forms the exponent from base e and the value of the operand "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

SIN: Form sine value

Description

The "Form sine value" instruction is used to calculate the sine of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is provided at output OUT and can be queried there.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form sine value":

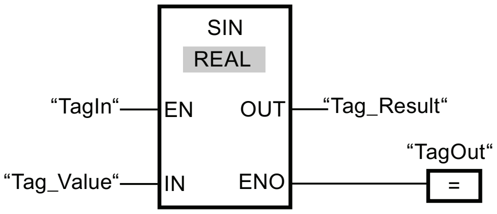
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Size of angle in the radian measure
OUT	Output	Floating-point numbers	I, Q, M, D, L	Sine of the specified angle

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796e+00 (π/2)
OUT	Tag_Result	1.0

If the operand "TagIn" has the signal state "1", the "Form sine value" instruction is executed. The instruction calculates the sine of the angle specified at input "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

COS: Form cosine value

Description

You can use the "Form cosine value" instruction to calculate the cosine of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is provided at output OUT and can be queried there.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form cosine value":

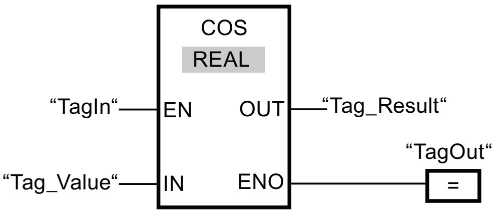
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Size of angle in the radian measure
OUT	Output	Floating-point numbers	I, Q, M, D, L	Cosine of the specified angle

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796e+00 (π/2)
OUT	Tag_Result	0

If the operand "TagIn" has the signal state "1", the "Form cosine value" instruction is executed. The instruction calculates the cosine of the angle specified at input "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

TAN: Form tangent value

Description

The "Form tangent value" instruction is used to calculate the tangent of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is provided at output OUT and can be queried there.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form tangent value":

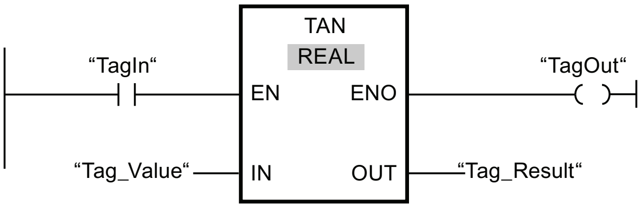
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Size of angle in the radian measure
OUT	Output	Floating-point numbers	I, Q, M, D, L	Tangent of the specified angle

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+3.141593e+00 (π)
OUT	Tag_Result	0

If the operand "TagIn" has the signal state "1", the "Form tangent value" instruction is executed. The instruction calculates the tangent of the angle specified at input "Tag_Value" and stores the result at output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

ASIN: Form arcsine value

Description

The "Form arcsine value" instruction is used to calculate the size of the angle from the sine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at input IN. The calculated angle size is given in the radian measure at output OUT and can range in value from -π/2 to +π/2.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is outside the permitted range (-1 to +1).

Parameters

The following table shows the parameters of the instruction "Form arcsine value":

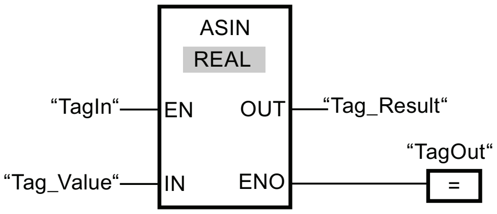
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Sine value
OUT	Output	Floating-point numbers	I, Q, M, D, L	Size of angle in the radian measure

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+1.570796e+00 (π/2)

If the operand "TagIn" has the signal state "1", the "Form arcsine value" instruction is executed. The instruction calculates the size of the angle corresponding to the sine value at input "Tag_Value". The result of the instruction is stored at output Tag_Result. If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

ACOS: Form arccosine value

Description

The "Form arccosine value" instruction is used to calculate the size of the angle from the arccosine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at input IN. The calculated angle size is given in the radian measure at output OUT and can range in value from 0 to +π.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is outside the permitted range (-1 to +1).

Parameters

The following table shows the parameters of the instruction "Form arccosine value":

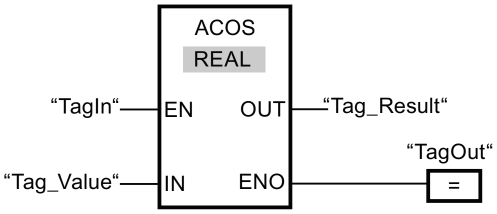
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Cosine value
OUT	Output	Floating-point numbers	I, Q, M, D, L	Size of angle in the radian measure

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	0
OUT	Tag_Result	+1.570796e+00 (π/2)

If the operand "TagIn" has the signal state "1", the "Form arccosine value" instruction is executed. The instruction calculates the size of the angle corresponding to the cosine value at input "Tag_Value". The result of the instruction is stored at output Tag_Result. If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

ATAN: Form arctangent value

Description

The "Form arctangent value" instruction is used to calculate the size of the angle from the arctangent value specified at input IN, which corresponds to this value. Only valid floating-point numbers may be specified at input IN. The calculated angle size is given in the radian measure at output OUT and can range in value from -π/2 to +π/2.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form arctangent value":

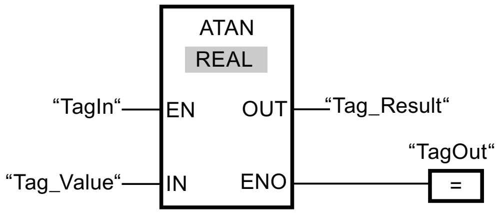
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Tangent value
OUT	Output	Floating-point numbers	I, Q, M, D, L	Size of angle in the radian measure

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+0.785398e+00 (π/4)

If the operand "TagIn" has the signal state "1", the "Form arctangent value" instruction is executed. The instruction calculates the size of the angle corresponding to the tangent value at input "Tag_Value". The result of the instruction is stored at output Tag_Result. If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

FRAC: Return fraction

Description

The "Return fraction" instruction is used to find the decimal places of the value at input IN. The result of the query is stored at output OUT and can be queried there. If the value at input IN is, for example, 123.4567, output OUT returns the value 0.4567. The instruction is started when the signal state at input EN is "1". In this case, the enable output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- Errors occur during the execution of the instruction, for example there is no valid floating-point number at the input.

Parameters

The following table shows the parameters of the "Return fraction" instruction:

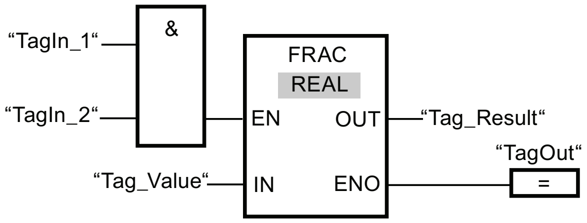
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value, whose decimal places are to be determined.
OUT	Output	Floating-point numbers	I, Q, M, D, L	Decimal places of the input value at input IN

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	2.555
OUT	Tag_Result	0.555

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Return fraction" instruction is executed. The decimal places from the value at the operand "Tag_Value" are copied to the operand "Tag_Result". If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

Adding additional inputs and outputs to FBD elements (Page 965)

EXPT: Exponentiate

Description

You can use the "Exponentiate" instruction to raise the value at the input IN1 by a power specified with the value at input IN2. The result of the instruction is stored at output OUT and can be queried there (OUT = IN1^{IN2}).

The value at input IN1 must be a valid floating-point number. Integers are also allowed for setting the input IN2.

Execution of the "Exponentiate" instruction requires a signal state of "1" at the EN enable input . In this case, the enable output ENO has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- Errors occur during the instruction processing, for example, if there is an overflow.

Parameters

The following table shows the parameters of the instruction "Exponentiate":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Floating-point numbers	I, Q, M, D, L or constant	Base value

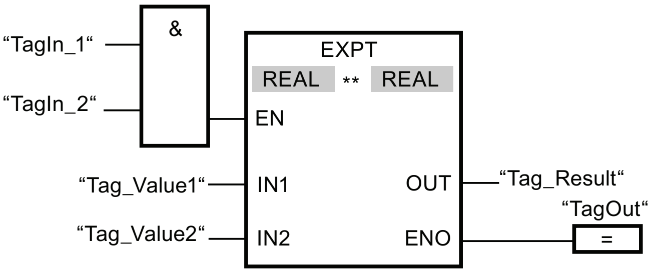
Parameters	Declaration	Data type	Memory area	Description
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Value with which the base value is exponentiated
OUT	Output	Floating-point numbers	I, Q, M, D, L	Result

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Exponentiate" instruction is executed. The value of operand "Tag_Value1" is raised by the power of the value of the operand "Tag_Value2". The result is stored at output "Tag_Result". If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

Adding additional inputs and outputs to FBD elements (Page 965)

Move operations

MOVE: Move value

Description

You use the "Move value" instruction to transfer the content of the operand at the IN input to the operand at the OUT1 output. The transfer is always made in the direction of ascending addresses.

The following table shows the possible transfers:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE ,TOD, CHAR
WORD	WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD, CHAR
DWORD	DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, CHAR
SINT	SINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
USINT	USINT, UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
INT	INT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UINT	UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
DINT	DINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UDINT	UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
REAL	REAL	DWORD, REAL
LREAL	LREAL	LREAL
TIME	TIME	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME
DATE	DATE	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, DATE
TOD	TOD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TOD
DTL	DTL	DTL
CHAR	CHAR	Character of a string BYTE, WORD, DWORD[1)]
Character of a string[1)]	Character of a string	Character of a string CHAR
ARRAY[2)]	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT

[1] You can also use the "Move value" instruction to transfer individual characters of a string (STRING) to operands of CHAR data type. The number of the character to be transferred is specified in square brackets beside the operand name. "MyString[2]", for example, transfers the second character of the "MyString" string. It is also possible to transfer from operands of the data type CHAR to the individual characters of a string. You can also replace a specific character of a string with a character of another string.

[2] Transferring entire arrays (ARRAY) is possible only when the array components of the operands at input IN and at output OUT1 are of the same data type.

If the bit length of the data type at input IN exceeds the bit length of the data type at output OUT1, the higher-order bits of the source value are lost. If the bit length of the data type at input IN is less than the bit length of the data type at output OUT1, the higher-order bits of the destination value will be overwritten with zeros.

In its initial state the instruction box contains 1 output (OUT1). The number of outputs can be extended. The added outputs are numbered in ascending order on the box. During the execution of the instruction the content of the operand at the input IN is transferred to all available outputs. The instruction box cannot be extended if structured data types (DTL, STRUCT, ARRAY) or characters of a string (STRING) are transferred.

The instruction is only executed if the signal state at the enable input EN is "1". In this case, enable output ENO also has the signal state "1".

If the signal state at enable input EN is "0", enable output ENO is reset to "0".

You can also use the "Move block" (MOVE_BLK) and "Move block uninterruptible" (UMOVE_BLK) instructions to copy operands of the ARRAY data type. You can copy operands of the STRING data type with the instruction S_MOVE.

Parameters

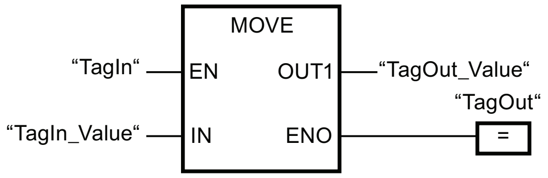
The following table shows the parameters of the "Move value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers, floating-point numbers, timers, DATE, TIME, TOD, DTL, CHAR, STRUCT, ARRAY	I, Q, M, D, L or constant	Element used to overwrite the destination address.
OUT1	Output	Bit strings, integers, floating-point numbers, timers, DATE, TIME, TOD, DTL, CHAR, STRUCT, ARRAY	I, Q, M, D, L	Destination address

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
OUT1	TagOut_Value	0011 1111 1010 1111

If the operand "TagIn" has the signal state "1", the "Move value" instruction is executed. The instruction copies the content of the operand "TagIn_Value" to the operand "TagOut_Value" and set output "TagOut" to signal state "1".

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

Removing instruction inputs and outputs (Page 966)

MOVE_BLK: Move block (Page 1501)

UMOVE_BLK: Move block uninterruptible (Page 1503)

S_MOVE: Move character string (Page 1678)

Adding additional inputs and outputs to FBD elements (Page 965)

FieldRead: Read field

Description

You can use the "Read field" instruction to read out a specific component fro the field specified at the parameter MEMBER and transfer its content to the tag at the parameter VALUE. You use the parameter INDEX to define the index of the field components that are to be read. At the parameter MEMBER you specify the first component of the field to be read.

The data types of the field component at parameter MEMBER and the tags at parameter VALUE must correspond to the data type of the instruction "Read field".

The execution of the instruction "Read field" is only started if the signal state at the enable input EN is "1". If no error occurs during execution, output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- The field component specified at the parameter INDEX is not defined in the field specified at the parameter MEMBER.
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Read field":

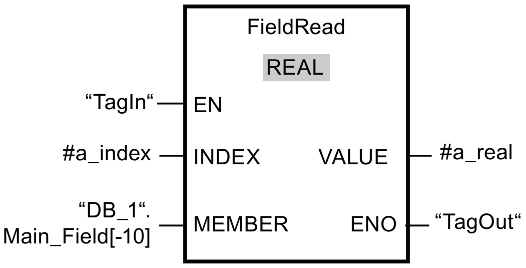
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
INDEX	Input	DINT	I, Q, M, D, L or constant	Index of field components whose content is read out
MEMBER	Input	BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR as components of a ARRAY tag	I, Q, M, D, L	First component of the field from which will be read
VALUE	Output	BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR	I, Q, M, D, L	Operand to which the contents of the field component are transferred.

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Tag	Value
INDEX	a_index	4
MEMBER	"DB_1".Main_Field[-10]	First component of the field "Main_Field[-10..10] of REAL" in the data block "DB_1"
VALUE	a_real	Component with Index 4 of the field "Main_Field[-10..10] of REAL"

The field component with index 4 is read out from the field "Main_Field[-10...10] of REAL" and written to the tag "a_real". The field component to be read is defined by the value at the parameter INDEX.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

FieldWrite: Write field

Description

The "Write field" instruction is used to transfer the content of the tag at the VALUE input to a specific component of the field at the MEMBER output. You use the value at the INDEX input to specify the index of the field component that is described. At the MEMBER output, enter the first component of the field which is to be written to.

The data types of the field component specified at the MEMBER output and the tags at the VALUE input have to match the data type of the "Write field" instruction.

The execution of the "Write field" instruction can only be started when the signal state at the enable input EN is "1". If no error occurs during execution, enable output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable Input EN has the signal state "0".
- The field component specified at the input INDEX is not defined in the field specified at the output MEMBER.
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Write field":

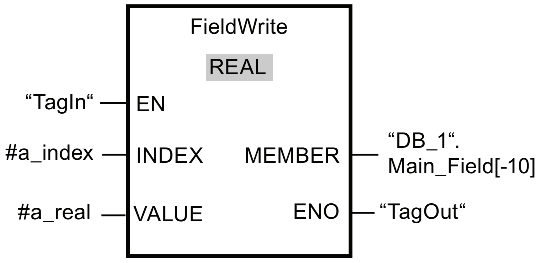
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
INDEX	Input	DINT	I, Q, M, D, L or constant	Index of field component that is written with the content of VALUE
VALUE	Input	BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR	I, Q, M, D, L or constant	Operand whose contents are copied
MEMBER	Output	BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR as components of a ARRAY tag	I, Q, M, D, L	First component of the field to which you write the content of VALUE

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
INDEX	a_index	4
VALUE	a_real	10,54
MEMBER	"DB_1".Main_Field[-10]	First component of the field "Main_Field[-10..10] of REAL" in the data block "DB_1"

The value 10.54 of the tag "a_real" is written in the field component with index 4 of the field "Main_Field[-10 ... 10] of REAL". The index of the field component to which the content of the "a_real" tag is transferred" is specified by the value at the INDEX input.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

MOVE_BLK: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The number of elements to be copied to the destination area is specified with the COUNT parameter. The width of the elements to be copied is defined by the width of the element at input IN. The copy operation takes place in the direction of ascending addresses.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- More data is copied than is made available at input IN or output OUT.

If the last BOOL element of an ARRAY structure is not located at a byte border (for example, bit 16 for 2 bytes), the ENO output remains set to "1" in the case of an overflow until the byte border of the ARRAY structure is exceeded. If the byte border of the ARRAY structure is exceeded by the value at input COUNT, output ENO is reset to "0".

Parameters

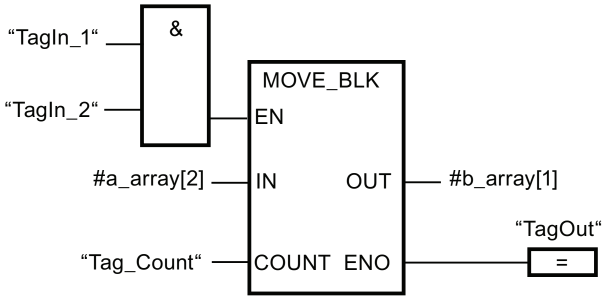
The following table shows the parameters of the "Move block" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Binary numbers, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	D, L	The first element of the source area to be copied.
COUNT	Input	UINT	I, Q, M, D, L or constant	Number of elements to be copied from the source area to the destination area.
OUT	Output	Binary numbers, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	D, L	The first element of the destination area to which the content of the source area is copied.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	a_array[2]	The operand "a_array" is an ARRAY data type and consists of 5 elements of the INT data type.
COUNT	Tag_Count	3
OUT	b_array[1]	The operand "b_array" is an ARRAY data type and consists of 6 elements of the INT data type.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Move block" instruction is executed. The instruction selects three INT elements from the tag "a_array" (a_array[2..4]) and copies their content to the output tag "b_array" (b_array[1..3]). If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

UMOVE_BLK: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to copy the contents of a memory area (source area) to another memory area (destination area) without the function being interrupted. The number of elements to be copied to the destination area is specified with the COUNT parameter. The width of the elements to be copied is defined by the width of the element at input IN. The content of the source area is copied to the destination area in the direction of the ascending address.

Note

The copy operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the "Move block uninterruptible" instruction.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- More data is copied than is made available at input IN or output OUT.

If the last BOOL element of an ARRAY structure is not located at a byte border (for example, bit 16 for 2 bytes), the ENO output remains set to "1" in the case of an overflow until the byte border of the ARRAY structure is exceeded. If the byte border of the ARRAY structure is exceeded by the value at input COUNT, output ENO is reset to "0".

Parameters

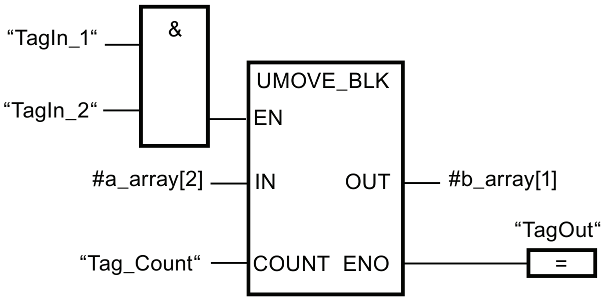
The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Binary numbers, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	D, L	The first element of the source area to be copied.
COUNT	Input	UINT	I, Q, M, D, L or constant	Number of elements to be copied from the source area to the destination area.
OUT	Output	Binary numbers, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	D, L	The first element of the destination area to which the content of the source area is copied.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	a_array[2]	The operand "a_array" is an ARRAY data type and consists of 5 elements of the INT data type.
COUNT	Tag_Count	3
OUT	b_array[1]	The tag "b_array" is an ARRAY data type and consists of 6 elements of the INT data type.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Move block uninterruptible" instruction is executed. The instruction selects three INT elements from the tag "a_array" (a_array[2..4]) and copies their content to the output tag "b_array" (b_array[1..3]). The copy operation cannot be interrupted by other operating system activities. If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

FILL_BLK: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the value of input IN. The destination area is filled beginning with the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at input IN is selected and copied to the destination area as often as specified by the value of the COUNT parameter. The copy operation takes place in the direction of ascending addresses.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- More data is copied than is made available at input IN or output OUT.

If the last BOOL element of an ARRAY structure is not located at a byte border (for example, bit 16 for 2 bytes), the ENO output remains set to "1" in the case of an overflow until the byte border of the ARRAY structure is exceeded. If the byte border of the ARRAY structure is exceeded by the value at input COUNT, output ENO is reset to "0".

Parameters

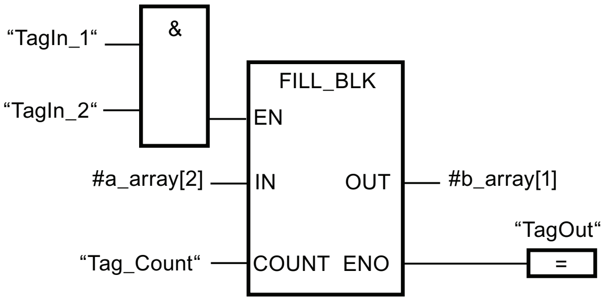
The following table shows the parameters of the "Fill block" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Binary numbers, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	D, L or constant	Element used to fill the destination area.
COUNT	Input	UINT	I, Q, M, D, L or constant	Number of repeated copy operations
OUT	Output	Binary numbers, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	D, L	Address in destination area where filling begins.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	A_array[2]	The operand "a_array" is an ARRAY data type and consists of 4 elements of the WORD data type (ARRAY[1..4] of WORD).
COUNT	Tag_Count	3
OUT	b_array[1]	The operand "b_array" is an ARRAY data type and consists of 5 elements of the WORD data type (ARRAY[1..5] of WORD).

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Fill block" instruction is executed. The instruction copies the second element (a_array[2]) of the tag "a_array" three times to the output tag "b_array" (b_array[1..3]). If no errors occur during the execution of the instruction, the outputs ENO and "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

UFILL_BLK: Fill block uninterruptible

Description

You can use the "Fill block uninterruptible" instruction to fill a memory area (destination area) uninterruptible with the value of input IN. The destination area is filled beginning with the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the input IN is selected and copied to the destination area as often as specified by the value of the COUNT parameter. The copy operation takes place in the direction of ascending addresses.

Note

The copy operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the "Fill block uninterruptible" instruction.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- More data is copied than is made available at input IN or output OUT.

If the last BOOL element of an ARRAY structure is not located at a byte border (for example, bit 16 for 2 bytes), the ENO output remains set to "1" in the case of an overflow until the byte border of the ARRAY structure is exceeded. If the byte border of the ARRAY structure is exceeded by the value at input COUNT, output ENO is reset to "0".

Parameters

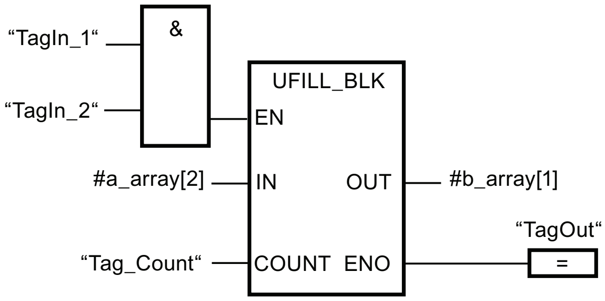
The following table shows the parameters of the "Fill block uninterruptible" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Binary numbers, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	D, L or constant	Element used to fill the destination area.
COUNT	Input	UINT	I, Q, M, D, L or constant	Number of repeated copy operations
OUT	Output	Binary numbers, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	D, L	Address in destination area where filling begins.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	a_array[2]	The operand "a_array" is an ARRAY data type and consists of 4 elements of the WORD data type (ARRAY[1..4] of WORD).
COUNT	Tag_Count	3
OUT	b_array[1]	The operand "b_array" is an ARRAY data type and consists of 5 elements of the WORD data type (ARRAY[1..5] of WORD).

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Fill block uninterruptible" instruction is executed. The instruction copies the second element (a_array[2]) of the tag "a_array" three times to the output tag "b_array" (b_array[1..3]). The copy operation cannot be interrupted by other operating system activities. If no errors occur during the execution of the instruction, the outputs ENO and "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 741)

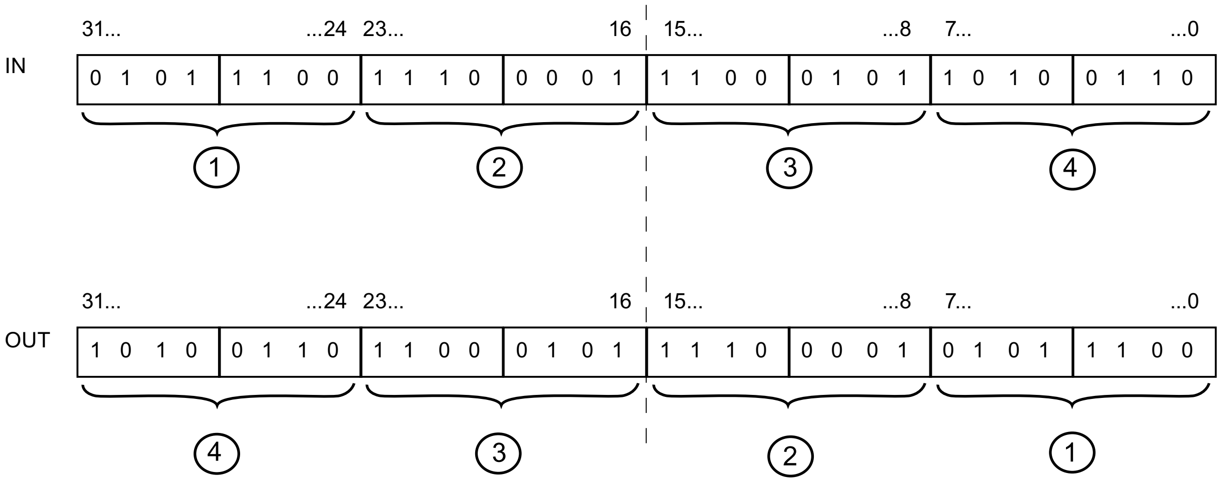
Basics of the EN/ENO mechanism (Page 819)

SWAP: Swap

Description

The "Swap" instruction is used to change the order of the bytes at input IN and query the result at output OUT.

The following figure shows how the bytes of a DWORD data type operand are swapped using the "Swap" instruction:



Execution of the "Swap" instruction requires a signal state of "1" at enable input EN. In this case, the enable output ENO has the signal state "1".

The enable output ENO is reset when the enable input EN has the signal state "0" or errors occur during the execution of the instruction.

Parameters

The following table shows the parameters of the "Swap" instruction:

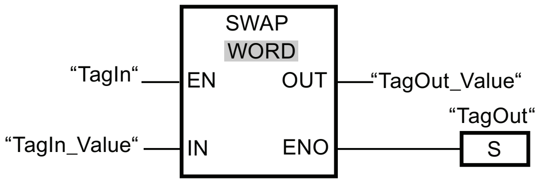
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD, DWORD	I, Q, M, D, L or constant	Operand whose bytes are swapped.
OUT	Output	WORD, DWORD	I, Q, M, D, L	Result

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0000 1111 0101 0101
OUT	TagOut_Value	0101 0101 1111 0000

If the operand "TagIn" has the signal state "1", the "Swap" instruction is executed. The arrangement of the bytes is changed and stored in the operand "TagOut_Value". If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

Conversion operations

CONVERT: Convert value

Description

The "Convert value" instruction reads the content of the IN parameter and converts it according to the data types configured in the instruction box. The converted value is provided at output OUT.

For information on possible conversions, refer to the "Explicit conversion (Page 799)" section

The execution of the "Convert value" instruction can only be started if the the signal state at the enable input EN is "1". If no error occurs during execution, output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.
- For the input IN, an operand of data type BYTE, WORD or DWORD is configured, whose highest value bit is set. A signed integer of data type SINT, INT or DINT is configured in the instruction box for the output OUT, which has the same bit length as the operand at input IN.

Parameters

The following table shows the parameters of the "Convert value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers, floating-point numbers, CHAR, BCD16, BCD32	I, Q, M, D, L or constant	Value to be converted.
OUT	Output	Bit strings, integers, floating-point numbers, CHAR, BCD16, BCD32	I, Q, M, D, L	Result of the conversion

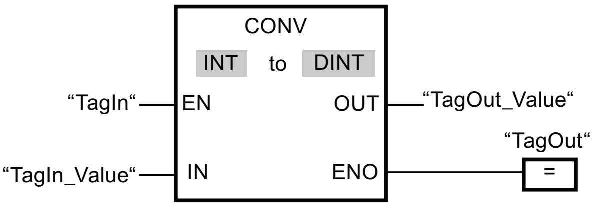
You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

Bit strings (BYTE, WORD, DWORD) cannot be selected in the instruction box. If you have specified an operand of data type BYTE, WORD or DWORD at a parameter of the instruction, the value of the operand is interpreted as an unsigned integer with the same bit length. In this case the data type BYTE is interpreted as USINT, WORD as UINT and DWORD as UDINT.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the content of the operand "TagIn_Value" is read and converted to an integer (16 bits). The result is stored in the operand "TagOut_Value". The output "TagOut" is set to "1" if the instruction was executed without errors.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

Explicit conversion (Page 799)

ROUND: Round numerical value

Description

The "Round numerical value" instruction is used to round the value at input IN to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts this to the nearest integer. If the input value is exactly between an even and odd number, then the even number will be converted. The result of the instruction is provided at output OUT and can be queried there.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, enable output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable Input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Round numerical value" instruction:

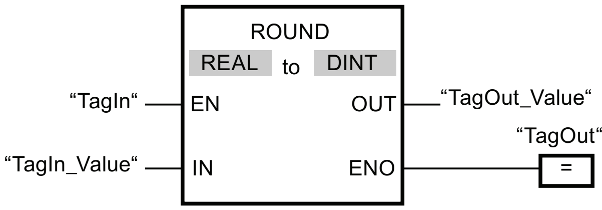
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value to be rounded.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result of the rounding

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	0	0

If the operand "TagIn" has the signal state "1", the "Round numerical value" instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the nearest even integer and sent to output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

CEIL: Generate next higher integer from floating-point number

Description

The instruction "Generate next higher integer from floating-point number" is used to round the value at input IN to the next higher integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next higher integer. The result of the instruction is provided at output OUT and can be queried there. The output value can be greater than or equal to the input value.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, enable output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable Input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Generate next higher integer from floating-point number":

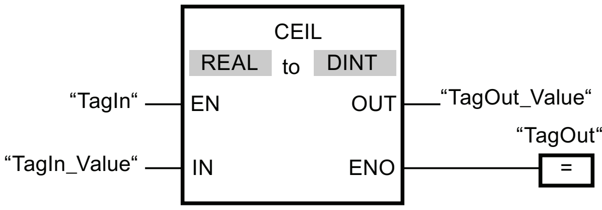
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value as floating-point number
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result with the next higher integer

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	1	0

If the operand "TagIn" has the signal state "1", the instruction "Generate next higher integer from floating-point number" is executed. The floating-point number at input "TagIn_Value" is rounded to the next higher integer and displayed at the output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

FLOOR: Generate next lower integer from floating-point number

Description

The instruction "Generate next lower integer from floating-point number" is used to round the value at input IN to the next lower integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next lower integer. The result of the instruction is provided at output OUT and can be queried there. The output value can be less than or equal to the input value.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, enable output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable Input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Generate next lower integer from floating-point number":

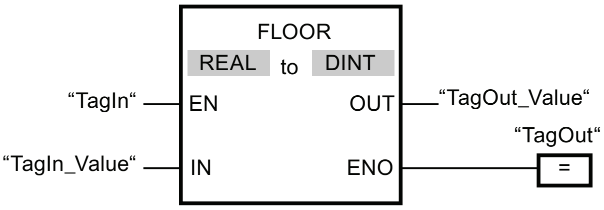
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value as floating-point number
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result with the next lower integer

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	0	-1

If the operand "TagIn" has the signal state "1", then the instruction "Generate next lower integer from floating-point number" will be executed. The floating-point number at input "TagIn_Value" is rounded to the next lower integer and displayed at output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

TRUNC: Truncate numerical value

Description

The "Truncate numerical value" instruction is used to form an integer from the value at input IN. The value at input IN is interpreted as a floating-point number. The instruction selects only the integer part of the floating-point number and sends this to output OUT without decimal places.

The instruction is only executed if the signal state at the enable input EN is "1". If no errors occur during the execution of the instruction, the output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Truncate numerical value":

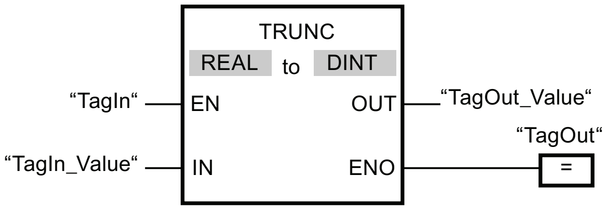
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value as floating-point number
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result with integer part of the floating-point number

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	0	0

If the operand "TagIn" has the signal state "1", the instruction "Truncate numerical value" is executed. The integer part of the floating-point number at input "TagIn_Value" is converted to an integer and sent to output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

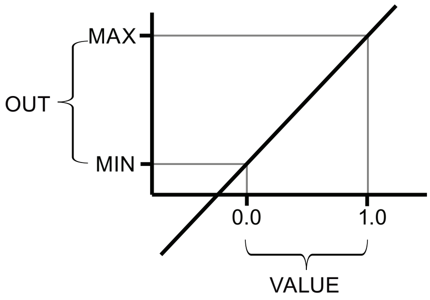
Basics of the EN/ENO mechanism (Page 819)

SCALE_X: Scale

Description

The "Scale" instruction is used to scale the value at input VALUE by mapping it to a specified value range. When the instruction "Scale" is executed, the floating-point value at input VALUE is scaled to the value range, which is defined by the parameters MIN and MAX. The result of the scaling is an integer, which is stored at output OUT.

The following figure shows an example of how values can be scaled:



Execution of the instruction "Scale" requires a signal state of "1" at enable input EN. In this case, the enable output ENO also has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- The value at input MIN is greater than or equal to the value at input MAX.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- An overflow occurs.
- The value at input VALUE is NaN (Not a number = result of an invalid arithmetic operation).

Parameters

The following table shows the parameters of the "Scale" instruction:

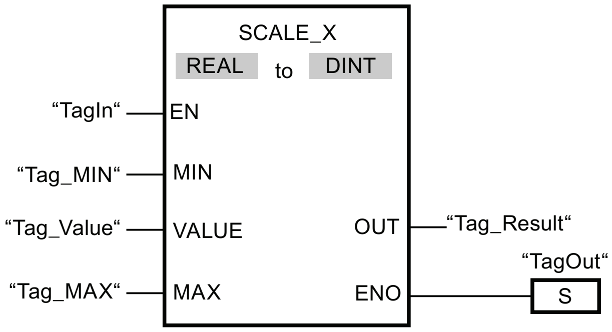
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VALUE	Input	Floating-point numbers	I, Q, M, D, L or constant	Value to be scaled.
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result of scaling

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
MIN	Tag_MIN	10
VALUE	Tag_Value	0.5
MAX	Tag_MAX	30
OUT	Tag_Result	20

If the operand "TagIn" has the signal state "1", the instruction "Scale" is executed. The value at input "Tag_Value" is scaled to the range of values defined by the values at inputs "Tag_MIN" and "Tag_MAX". The result is stored at output "Tag_Result". If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

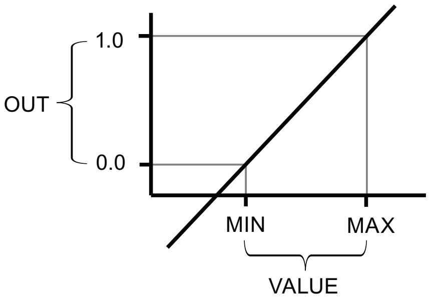
NORM_X: Normalize (Page 1522)

NORM_X: Normalize

Description

The "Normalize" instruction is used to normalize the value of the tag at input VALUE by mapping it to a linear scale. You can use the parameters MIN and MAX to define the limits of a value range that is applied to the scale. Depending on the location of the normalized value in this value range, the result at output OUT is calculated and stored as a floating-point number. If the value to be normalized is equal to the value at input MIN, output OUT returns the value "0.0". If the value to be normalized is equal to the value at input MAX, output OUT has the value "1.0".

The following figure shows an example of how values can be normalized:



Execution of the instruction "Normalize" requires the signal state of "1" at enable input EN. In this case, the enable output ENO has the signal state "1".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Input EN has the signal state "0".
- The value at input MIN is greater than or equal to the value at input MAX.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- The value at input VALUE is NaN (result of an invalid arithmetic operation).

Parameters

The following table shows the parameters of the instruction "Normalize":

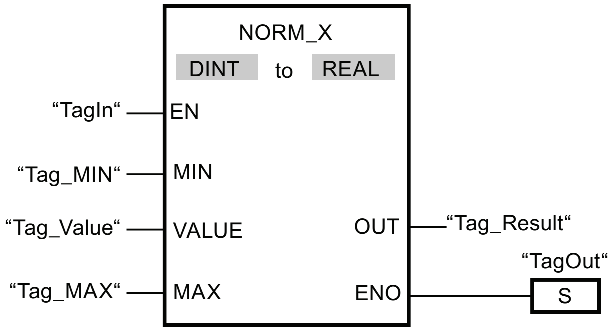
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VALUE	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Value to be normalized.
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
OUT	Output	Floating-point numbers	I, Q, M, D, L	Result of the normalization

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
MIN	Tag_MIN	10
VALUE	Tag_Value	20
MAX	Tag_MAX	30
OUT	Tag_Result	0.5

If the operand "TagIn" has the signal state "1", the instruction "Normalize" is executed. The value at input "Tag_Value" is assigned to the range of values defined by the values at inputs "Tag_MIN" and "Tag_MAX". The tag value at input "Tag_Value" is normalized corresponding to the defined value range. The result is stored as a floating-point number at output "Tag_Result". If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

SCALE_X: Scale (Page 1520)

Program control operations

JMP: Jump if RLO = 1

Description

The instruction "Jump if RLO = 1" is used to interrupt the linear execution of the program and resume it in another network. The target network must be identified by a jump label (LABEL). The jump label description is entered in the placeholder above the instruction box.

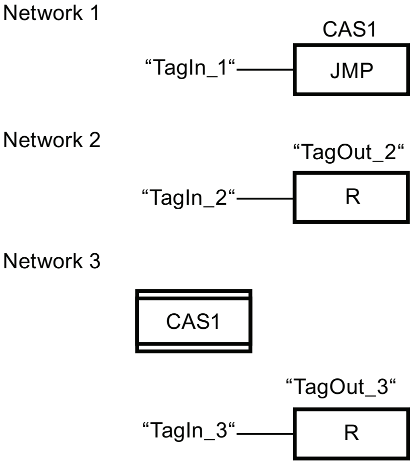
The jump label must be in the same block in which the instruction is executed. The name of a jump label must be unique and can only be assigned once in a block.

If the result of logic operation (RLO) at the input of the instruction is "1", the jump to the network identified by the jump label is executed. The jump direction can be towards higher or lower network numbers.

If the condition at the input of the instruction is not fulfilled (RLO = 0), the linear execution of the program is not interrupted but instead continues in the next network.

Example

The following example shows how the instruction works:



If the operand "TagIn_1" has the signal state "1", the instruction "Jump if RLO = 1" is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If input "TagIn_3" has the signal state "1", output "TagOut_3" is reset.

See also

Overview of the valid data types (Page 741)

JMPN: Jump if RLO = 0

Description

You can use the instruction "Jump if RLO = 0" to interrupt the linear execution of the program and resume it in another network, when the result of logic operation at the input of the instruction is "0". The target network must be identified by a jump label (LABEL). The jump label description is entered in the placeholder above the instruction box.

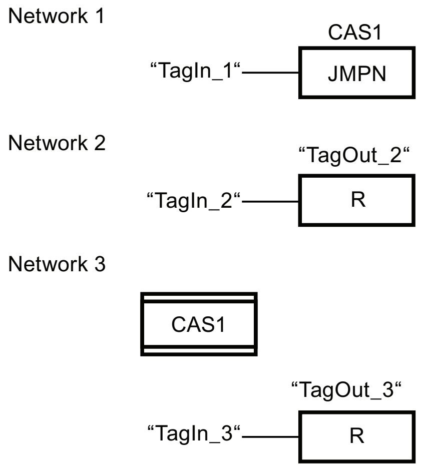
The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block.

If the result of logic operation (RLO) at the input of the instruction is "0", the jump to the network identified by the specified jump label is executed. The jump direction can be towards higher or lower network numbers.

If the result of the logic operation at the input of the instruction is "1", execution of the program continues in the next network.

Example

The following example shows how the instruction works:



If the operand "TagIn_1" has the signal state "0", the instruction "Jump if RLO = 0" is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If input "TagIn_3" has the signal state "1", output "TagOut_3" is reset.

See also

Overview of the valid data types (Page 741)

LABEL: Jump label

Description

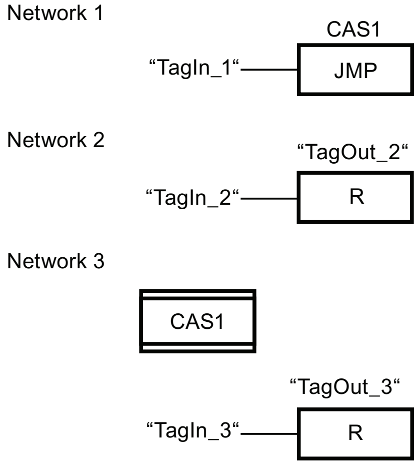
The jump label identifies a destination network in which the exeuction of the program can be resumed after the execution of a jump instruction.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block.

Only one jump label can be placed in a network. Each jump label can jump to several locations.

Example

The following example shows how the instruction works:



If the operand "TagIn_1" has the signal state "1", the instruction "Jump if RLO = 1" is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If input "TagIn_3" has the signal state "1", output "TagOut_3" is reset.

JMP_LIST: Define jump list

Description

The instruction "Define jump list" is used to define several conditional jumps and to resume the program execution in a defined network depending on the value of the parameter K.

The jumps are defined by jump labels ((LABEL)), which you specify at the outputs of the instruction box. In its initial state the instruction box contains at least 2 outputs (DEST0 and DEST1). The number of outputs can be extended. The numbering of the outputs begins with the value "0" and is continued in ascending order with each new output. Only jump labels can be specified at the outputs of the instruction. It is not permitted to specify instructions or operands.

You use the value of the parameter K to specify the number of the output and, accordingly, the jump label at which the program execution will be continued. If the value at the parameter K is greater than the number of available outputs, the linear execution of the program is not interrupted but instead continued in the next network.

The instruction "Define jump list" is only executed if the signal state at the enable input EN is "1".

Parameters

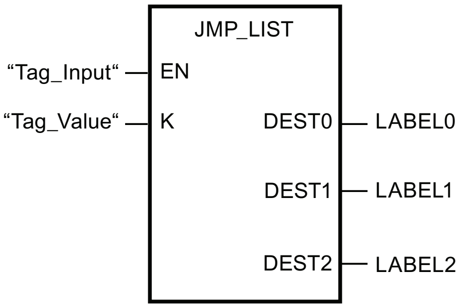
The following table shows the parameters of the instruction "Define jump list":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, L, D	Enable input
K	Input	UINT	I, Q, M, L, D or constant	Specifies the number of the output and thus the jump that will be executed. (K=0 to 99)
DEST0	-	-	-	First jump label
DEST1	-	-	-	Second jump label
DESTn	-	-	-	Optional jump labels (n = 2 to 99)

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand/Jump label	Value
K	"Tag_Value"	1
DEST0	LABEL0	Jump in the network that is identified with the jump label "LABEL0".
DEST1	LABEL1	Jump in the network that is identified with the jump label "LABEL1".
DEST2	LABEL2	Jump in the network that is identified with the jump label "LABEL2".

If the operand "Tag_Input" has the signal state "1", the instruction "Define jump list" is executed. The execution of the program is continued according to the value of the operand "Tag_Value" in the network that is identified with the jump label "LABEL1".

See also

Overview of the valid data types (Page 741)

Adding additional inputs and outputs to FBD elements (Page 965)

Removing instruction inputs and outputs (Page 966)

SWITCH: Jump distributor

Description

The instruction "Jump distributor" is used to define several program jumps, which are executed depending on the result of one or more comparison instructions.

At the parameter K, you specify the value to be compared. This value is compared with the values that the individual inputs return. You select the type of comparison for each input. The availability of various comparison instructions depends on the data type of the instruction.

The following table shows the comparison instructions that are available depending on the selected data type:

Data type	Instruction	Syntax
Bit strings	Equal	==
	Not equal	<>
Integers, floating-point numbers, TIME, DATE, TOD	Equal	==
	Not equal	<>
	Greater or equal	>=
	Less or equal	<=
	Greater than	>
	Less than	<

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box. If you select a comparison instruction and the data type of the instruction is not yet defined, then the "<???>" drop-down list will only list those data types that are permitted for the selected comparison instruction.

The execution of the instruction begins with the first comparison and is executed until a comparison condition is fulfilled. When a comparison condition is fulfilled the subsequent comparison conditions are not considered. If none of the specified comparison conditions are fulfilled, the jump is executed at output ELSE. If not jump label is defined at output ELSE, the linear execution of the program is not interrupted but instead continued in the next network.

In its initial state the instruction box contains at least 2 outputs (DEST0 and DEST1). The number of outputs can be extended. The numbering of the outputs begins with the value "0" and is continued in ascending order with each new output. Specify jump labels (LABEL) at the outputs of the instruction. It is not permitted to specify instructions or operands at the outputs of the instruction.

An input is automatically inserted to each additional output. The jump programmed at an output is executed when the comparison condition of the corresponding is fulfilled.

Parameters

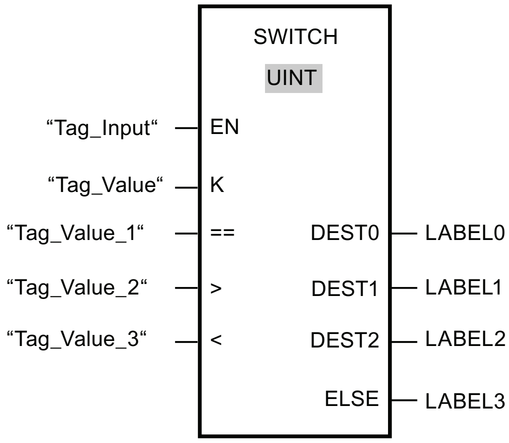
The following table shows the parameters of the instruction "Jump distributor":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
K	Input	UINT	I, Q, M, D, L or constant	Specifies the value to be compared.
<Comparison values>	Input	Bit strings, integers, floating-point numbers, TIME, DATE, TOD	I, Q, M, D, L or constant	Input values with which the value of the parameter K is compared.
DEST0	-	-	-	First jump label
DEST1	-	-	-	Second jump label
DEST(n)	-	-	-	Optional jump labels (n = 2 to 99)
ELSE	-	-	-	Program jump which is executed if none of the comparison conditions are fulfilled.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand/Jump label	Value
K	Tag_Value	23
==	Tag_ Value_1	20
>	Tag_ Value_2	21
<	Tag_ Value_3	19
DEST0	LABEL0	Jump to jump label "LABEL0", if the value of parameter K is equal to 20.
DEST1	LABEL1	Jump to jump label "LABEL1", if the value of parameter K is greater than 21.
DEST2	LABEL2	Jump to jump label "LABEL2", if the value of parameter K is less than 19.
ELSE	LABEL 3	Jump to jump label "LABEL3", if none of the comparison conditions are fulfilled.

If the operand "Tag_Input" changes to signal state "1", the instruction "Jump distributor" is executed. The execution of the program is continued in the network that is identified with the jump label "LABEL1".

See also

Overview of the valid data types (Page 741)

Adding additional inputs and outputs to FBD elements (Page 965)

Removing instruction inputs and outputs (Page 966)

RET: Return

Description

You can use the instruction "Return" to stop the execution of a block. This results in three types, in which the block processing can be completed:

- Without call of the "Return" instruction

  The block is exited after the execution of the last network. The ENO of the call function is set to the signal state "1".

- Call of the "Return" instruction with logic operation (see example)

  If the left connector has the signal state "1", then the block will be exited. The ENO of the function call corresponds to the operand.

- Call of the "Return" instruction without logic operation

  The block will be exited. The ENO of the function call corresponds to the operand.

Note

Only one jumping coil could be used in a network ("Return", "Jump if RLO=1", "Jump if RLO=0").

If this the result of logic operation (RLO) at the input of the instruction "Return" is "1", the execution of the program is terminated in the currently called block and continued after the call function in the calling block (for example, in the calling OB). The status (ENO) of the call function is determined by the parameter of the instruction. This can assume the following values:

- RLO
- TRUE/FALSE
- <Operand>

To set the parameter values, double-click the instruction and select the corresponding value in the drop-down list.

The following table shows the status of the call function when the instruction "Return" is programmed in a network within the called block:

RLO	Parameter value	ENO of the call function
1	RLO	1
	TRUE	1
	FALSE	0
	<Operand>	<Operand>
0	RLO	In this case, the execution of the program continues in the next network of the called block.
	TRUE	
	FALSE	
	<Operand>	

If an OB is completed, another block will be selected by the priority class system and started or re-executed.

- If the OB program cycle was completed, it will be restarted.
- If an OB, which interrupted another block (e.g. an Alarm OB), is completed, then the interrupted block (e.g. OB program cycle) will be executed.

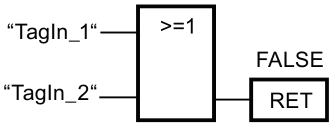
Parameters

The following table shows the parameters of the instruction "Return":

Parameters	Declaration	Data type	Memory area	Description
RLO	-	-	-	The status of the call function is set to the signal state of the RLO.
TRUE	-	-	-	When RLO=1, the status of the call function is set to "1".
FALSE	-	-	-	When RLO=1, the status of the call function is set to "0".
<Operand>	Input	BOOL	I, Q, M, D, L	When RLO=1, the status of the call function is set to the signal state of the specified operand.

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "TagIn_2" have signal state "1", the instruction "Return" is executed. Program execution in the called block is terminated and continues in the calling block. Output ENO of the call function is reset to signal state "0".

See also

Overview of the valid data types (Page 741)

Runtime control

RE_TRIGR: Restart cycle monitoring time

Description

The instruction "Restart cycle monitoring time" is used to restart the cycle monitoring of the CPU. The maximum cycle time then restarts with the time you set in the CPU configuration. By restarting the maximum cycle time, you can prevent errors being triggered or the CPU changing to STOP.

The instruction "Restart cycle monitoring time" can be used in blocks of the priority class 1 (cyclic OBs) and in the blocks called within it.

If the instruction is called in a block with a higher priority, for example a process interrupt, a diagnostics interrupt or a cyclic interrupt, the instruction is not executed and the enable output ENO is set to signal state "0".

Parameters

The instruction "Restart cycle monitoring time" has no parameters.

STP: Exit program

Description

The instruction "Exit program" instruction is used to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP state depend on the CPU configuration.

When the RLO at the input of the instruction is "1", the CPU changes to STOP mode and the execution of the program is terminated. The signal state at the output of the instruction is not evaluated.

If the RLO at the input of the instruction is "0", then the instruction will not be executed.

Parameters

The instruction "Exit program" has no parameters.

GetError: Get error locally

Description

The "Get error locally" instruction is used to query the occurrence of errors within a block. If the system signals errors during block execution, detailed information about the first error that occurred is saved in the operand at output ERROR.

Only operands of the system data type "ErrorStruct" can be specified at output ERROR. The system data type "ErrorStruct" specifies the exact structure in which the information about the error is stored. Using additional instructions, you can evaluate this structure and program an appropriate response. When the first error has been eliminated, the instruction outputs information about the next error that occurred.

Parameters

The following table shows the parameters of the instruction "Get error locally":

Parameters	Declaration	Data type	Memory area	Description
ERROR	Output	ErrorStruct	D, L	Error information

"ErrorStruct" data type

The following table shows the structure of the data type ErrorStruct:

<table>
<tr><th colspan="2">Structure components</th><th>Data type</th><th colspan="6">Description</th></tr>
<tr><td colspan="2">ERROR_ID</td><td>WORD</td><td colspan="6">Error ID</td></tr>
<tr><td colspan="2">FLAGS</td><td>BYTE</td><td colspan="6">Shows if an error occurred during a block call.
16#01: Error during a block call.
16#00: No error during a block call.</td></tr>
<tr><td colspan="2">REACTION</td><td>BYTE</td><td colspan="6">Default reaction:
0: Ignore (write error),
1: Continue with substitute value "0" (read error),
2: Skip instruction (system error)</td></tr>
<tr><td colspan="2">CODE_ADDRESS</td><td>CREF</td><td colspan="6">Information on address and type of block</td></tr>
<tr><td></td><td>BLOCK_TYPE</td><td>BYTE</td><td colspan="6">Type of block where the error occurred:
1: OB
2: FC
3: FB</td></tr>
<tr><td></td><td>CB_NUMBER</td><td>UINT</td><td colspan="6">Number of the code block</td></tr>
<tr><td></td><td>OFFSET</td><td>UDINT</td><td colspan="6">Reference to the internal memory</td></tr>
<tr><td colspan="2" rowspan="2">MODE</td><td rowspan="2">BYTE</td><td colspan="6">Access mode: Depending on the type of access, the following information can be output:</td></tr>
<tr><td>Mode</td><td>(A)</td><td>(B)</td><td>(C)</td><td>(D)</td><td>(E)</td></tr>
</table>

Structure components		Data type	Description					
			0					
			1					Offset
			2			Area		
			3	Location	Scope		Number	
			4			Area		Offset
			5			Area	DB no.	Offset
			6	PtrNo./Acc		Area	DB no.	Offset
			7	PtrNo./Acc	Slot No. / Scope	Area	DB no.	Offset
OPERAND_NUMBER		UINT	Operand number of the machine command					
POINTER_NUMBER_ LOCATION		UINT	(A) Internal pointer					
SLOT_NUMBER_SCOPE		UINT	(B) Storage area in internal memory					
DATA_ADDRESS		NREF	Information about the address of an operand					
	AREA	BYTE	(C) Memory area: L: 16#40 – 4E, 86, 87, 8E, 8F, C0 – CE I: 16#81 Q: 16#82 M: 16#83 DB: 16#84, 85, 8A, 8B					
	DB_NUMBER	UINT	(D) Number of the data block					
	OFFSET	UDINT	(E) Relative address of the operand					

Structure components "ERROR_ID"

The following table shows the values that can be output on the structure components "ERROR_ID":

ID (hexadecimal)	ID (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2505	9477	Calling the instruction "Stop" (SFC46) in the user program
2520	9504	Invalid STRING
2522	9506	Read errors: Operand outside the valid range
2523	9507	Write errors: Operand outside the valid range
2524	9508	Read errors: Invalid operand
2525	9509	Write errors: Invalid operand

ID (hexadecimal)	ID (decimal)	Description
2528	9512	Read errors: Data alignment
2529	9513	Write errors: Data alignment
252C	9516	Invalid pointer
2530	9520	Write errors: Data block
2533	9523	Invalid pointer used
2534	9524	Block number error FC
2535	9525	Block number error FB
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2942	10562	Read errors: Input
2943	10563	Write errors: Output

The enable output ENO of the instruction "Get error locally" instruction is set only if the enable input EN returns signal state "1" and error information is present. If one of these conditions does not apply, then the remaining program execution is not affected by the instruction "Get error locally".

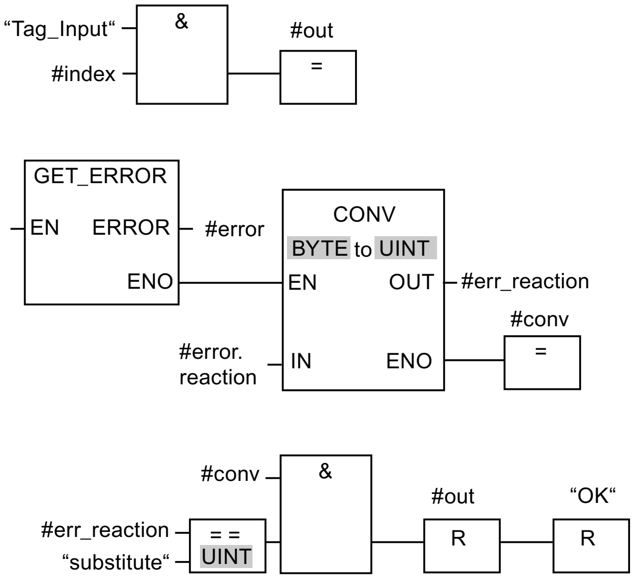
The instruction "Get error locally" can also be used to forward an alarm about the error status to the calling block. To do this, the instruction must be positioned in the last network of the called block.

Note

The instruction "Get error locally" enables local error handling within a block. When "Get error locally" is inserted in the program code of a block, any predefined system responses are ignored if an error occurs.

Example

The following example shows how the instruction works:



If an error occurs, the instruction "Get error locally" returns the error information to the locally created structure "#error" at output ERROR. The error information is converted and evaluated with the comparison instruction "Equal to". Information about the type of error is the first comparison value assigned to the instruction. For the second comparison value, a value of "1" is specified in the operand "substitute". If the error is a read error, the condition of the comparison instruction is satisfied. In this case the outputs "#out" and "OK" are reset.

See also

Overview of the valid data types (Page 741)

Basics of error handling (Page 1009)

Principles of local error handling (Page 1011)

Error output priorities (Page 1012)

Enabling local error handling for a block (Page 1013)

GetErrorID: Get error ID locally

Description

The "Get error ID locally" instruction is used to query the occurrence of errors within a block. If the system signals errors during block execution, the error ID of the first error that occurred is saved in the tag at output ID. Only tags of the WORD data type can be specified at the ID output. When the first error has been eliminated, the instruction outputs the error ID of the next error that occurred.

The output of the instruction "Get error ID locally" is only set if the input of the instruction returns signal state "1" and error information is present. If one of these conditions does not apply, then the remaining program execution is not affected by the instruction "Get error ID locally".

The instruction "Get error ID locally" can also be used to forward an alarm about the error status to the calling block. To do this, the instruction must be positioned in the last network of the called block.

Note

The instruction "Get error ID locally" enables local error handling within a block. When "Get error ID locally" is inserted in the program code of a block, any predefined system responses are ignored if an error occurs.

Parameters

The following table shows the parameters of the instruction "Get error ID locally":

Parameters	Declaration	Data type	Memory area	Description
ID	Output	WORD	I, Q, M, D, L	Error ID

Parameter ID

The following table shows the values that can be output at the parameter ID:

ID (hexadecimal)	ID (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2505	9477	Calling the instruction "Stop" (SFC46) in the user program
2520	9504	Invalid STRING
2522	9506	Read errors: Operand outside the valid range
2523	9507	Write errors: Operand outside the valid range
2524	9508	Read errors: Invalid operand
2525	9509	Write errors: Invalid operand

ID (hexadecimal)	ID (decimal)	Description
2528	9512	Read errors: Data alignment
2529	9513	Write errors: Data alignment
252C	9516	Invalid pointer
2530	9520	Write errors: Data block
2533	9523	Invalid pointer used
2534	9524	Block number error FC
2535	9525	Block number error FB
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2942	10562	Read errors: Input
2943	10563	Write errors: Output

See also

Overview of the valid data types (Page 741)

Basics of error handling (Page 1009)

Principles of local error handling (Page 1011)

Error output priorities (Page 1012)

Enabling local error handling for a block (Page 1013)

Word logic operations

AND: AND logic operation

Description

The instruction "AND logic operation" is used to link the value at input IN1 to the value at input IN2 bit-by-bit by AND logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by AND logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified values.

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box. During the execution of the instruction, the values of all available input parameters are linked by AND logic. The result is stored at output "OUT".

The result bit has the signal state "1" only when both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has the signal state "0", the corresponding result bit is reset.

The instruction is only executed if the signal state at the enable input EN is "1". In this case, output ENO also has the signal state "1".

If the signal state at the enable input EN is "0", the signal state at the enable output ENO is also "0".

Parameters

The following table shows the parameters of the instruction "AND logic operation":

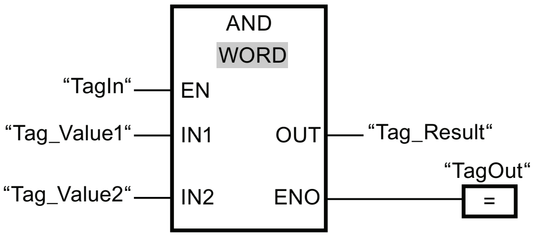
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L or constant	Optional input values
OUT	Output	Bit strings	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	01010101 01010101
IN2	Tag_Value2	00000000 00001111
OUT	Tag_Result	00000000 00000101

If the operand "TagIn" has the signal state "1", the instruction "AND logic operation" is executed. The value of operand "Tag_Value1" is linked by AND to the value of the operand "Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". Output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 741)

Adding additional inputs and outputs to FBD elements (Page 965)

Removing instruction inputs and outputs (Page 966)

Basics of the EN/ENO mechanism (Page 819)

OR: OR logic operation

Description

The instruction "OR logic operation" is used to link the value at input IN1 to the value at input IN2 bit-by-bit by OR logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by OR logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all bits of the specified tags.

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended in the instruction box. The inserted inputs are numbered in ascending order in the box. During the execution of the instruction, the values of all available input parameters are linked by OR logic. The result is stored at output "OUT".

The result bit has the signal state "1" when at least one of the two bits in the logic operation has the signal state "1". If both of the bits of the logic operation have signal state "0", the corresponding result bit is reset.

The instruction is only executed if the signal state at the enable input EN is "1". In this case, output ENO also has the signal state "1".

If the signal state at the enable input EN is "0", the signal state at the enable output ENO is also "0".

Parameters

The following table shows the parameters of the instruction "OR logic operation":

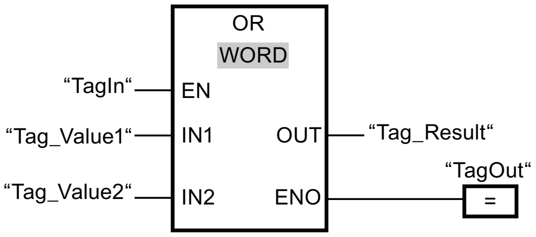
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L or constant	Optional input values
OUT	Output	Bit strings	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	01010101 01010101
IN2	Tag_Value2	00000000 00001111
OUT	Tag_Result	01010101 01011111

If the operand "TagIn" has the signal state "1", the instruction "OR logic operation" is executed. The value of operand "Tag_Value1" is linked by OR to the value of the operand "Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". Output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 741)

Adding additional inputs and outputs to FBD elements (Page 965)

Removing instruction inputs and outputs (Page 966)

Basics of the EN/ENO mechanism (Page 819)

XOR: EXCLUSIVE OR logic operation

Description

The instruction "EXCLUSIVE OR logic operation" is used to link the value at input IN1 to the value at input IN2 bit-by-bit by EXCLUSIVE OR logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by EXCLUSIVE OR logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified value.

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended in the instruction box. The inserted inputs are numbered in ascending order in the box. During the execution of the instruction, the values of all available input parameters are linked by EXCLUSIVE OR logic. The result is stored at output "OUT".

The result bit has the signal state "1" when one of the two bits in the logic operation has the signal state "1". If both of the bits of the logic operation have signal state "1" or "0", the corresponding result bit is reset.

The instruction is only executed if the signal state at the enable input EN is "1". In this case, output ENO also has the signal state "1".

If the signal state at the enable input EN is "0", the signal state at the enable output ENO is also "0".

Parameters

The following table shows the parameters of the instruction "EXCLUSIVE OR logic operation":

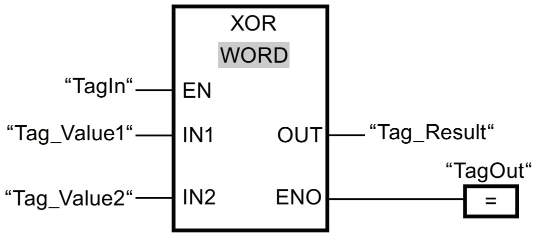
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L or constant	Optional input values
OUT	Output	Bit strings	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	01010101 01010101
IN2	Tag_Value2	00000000 00001111
OUT	Tag_Result	01010101 01011010

If the operand "TagIn" has the signal state "1", the instruction "EXCLUSIVE OR logic operation" is executed. The value of operand "Tag_Value1" is linked by EXCLUSIVE OR to the value of the operand "Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". Output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 741)

Adding additional inputs and outputs to FBD elements (Page 965)

Removing instruction inputs and outputs (Page 966)

Basics of the EN/ENO mechanism (Page 819)

INV: Create ones complement

Description

You can use the instruction "Create ones complement" to invert the signal status of the bits at input IN. When the instruction is processed, the value at input IN is linked to EXCLUSIVE OR by a hexadecimal mask (W#16#FFFF for 16-bit numbers or DW#16#FFFF FFFF for 32-bit number). This inverts the signal state of the individual bits that are then stored at output OUT.

The instruction is only executed if the signal state at the enable input EN is "1". In this case, output ENO also has the signal state "1".

If the signal state at the enable input EN is "0", the signal state at the enable output ENO is also "0".

Parameters

The following table shows the parameters of the instruction "Create ones complement":

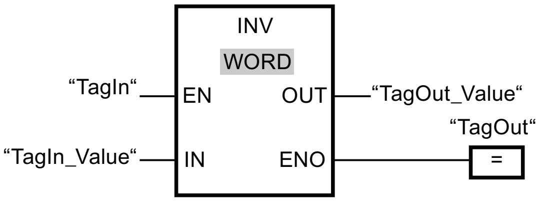
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	I, Q, M, D, L or constant	Input value
OUT	Output	Bit strings, integers	I, Q, M, D, L	Ones complement of the value at input IN

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	W#16#000F	W#16#7E
OUT	TagOut_Value	W#16#FFF0	W#16#81

If the operand "TagIn" has the signal state "1", then the instruction "Truncate numerical value" will be executed. The instruction inverts the signal state of the individual bits at input "TagIn_Value" and writes the result to output "TagOut_Value". Output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

DECO: Decode

Description

The "Decode" instruction is used to set a bit specified by the input value in the output value.

The instruction "Decode" reads the value at input IN and sets the bit in the output value, whose bit position corresponds to the read value. The other bits in the output value will be overwritten with zeroes. If the value at input IN is greater than 31, a modulo 32 instruction is executed.

The instruction "Decode" is only started if the signal state at the enable input EN is "1". If no error occurs during execution, output ENO also has the signal state "1".

If the signal state at the enable input EN is "0", the signal state at the enable output ENO is also "0".

Parameters

The following table shows the parameters of the "Decode" instruction:

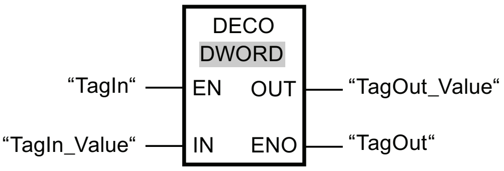
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	UINT	I, Q, M, D, L or constant	Input value
OUT	Output	Bit strings	I, Q, M, D, L	Output value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

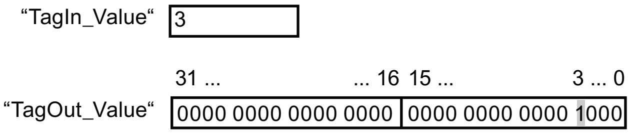
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following figure shows how the instruction works using specific operand values:



If the operand "TagIn" has the signal state "1", the "Decode" instruction is executed. The instruction interprets the value at input "TagIn_Value" as bit position "3" and sets the third bit to the value at output "TagOut_Value".

If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

ENCO: Encode

Description

The instruction "Encode" is used to read the bit number of the lowest bit in the input value and output it to the output OUT.

The "Encode" instruction selects the least significant bit of the value at the IN input and writes its bit number to the tag in the OUT output.

The "Encode" instruction is only started when the signal state at the EN enable input is "1". If no error occurs during execution, output ENO also has the signal state "1".

If the signal state at the enable input EN is "0", the signal state at the enable output ENO is also "0".

Parameters

The following table shows the parameters of the "Encode" instruction:

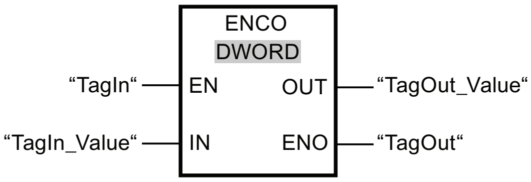
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings	I, Q, M, D, L or constant	Input value
OUT	Output	INT	I, Q, M, D, L	Output value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

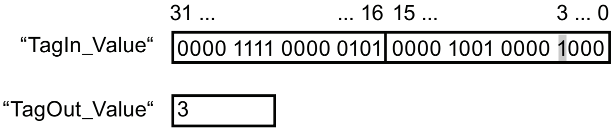
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following figure shows how the instruction works using specific operand values:



If operand "TagIn" has the signal state "1", the "Encode" instruction is executed. The instruction selects bit position "3" as the least significant bit at input "TagIn_Value" and writes the value "3" to the tag at the output "TagOut_Value.

If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

SEL: Select

Description

Depending on the signal state at switch (input G), the "Select" instruction selects one of the inputs IN0 or IN1 and moves its content to the output OUT. When the input G has the signal state "0", the value at the input IN0 is moved. When the input G has the signal state "1", the value at the input IN1 is moved to the output OUT.

The instruction can only be executed if the enable input EN has the signal state "1" and the tags at all parameters are of the same data type. If no errors occur during the execution of the instruction, enable output ENO also has the signal state "1".

The enable output ENO is reset when the enable input EN has the signal state "0" or errors occur during the execution of the instruction.

Parameters

The following table shows the parameters of the "Select" instruction:

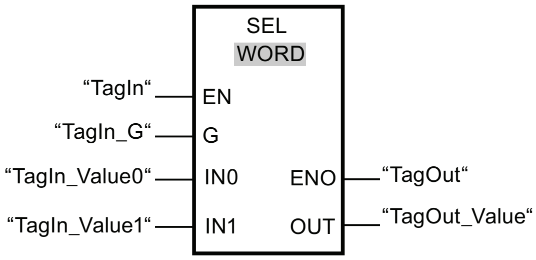
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
G	Input	BOOL	I, Q, M, D, L	Switch
IN0	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, character	I, Q, M, D, L or constant	First input value
IN1	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, character	I, Q, M, D, L or constant	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, character	I, Q, M, D, L	Result

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
G	TagIn_G	0	1
IN0	TagIn_Value0	W#16#0000	W#16#4C
IN1	TagIn_Value1	W#16#FFFF	W#16#5E
OUT	TagOut_Value	W#16#0000	W#16#5E

If the operand "TagIn" has the signal state "1", the instruction "Select" is executed. Depending on the signal state at input "TagIn_G", the value at input "TagIn_Value0" or "TagIn_Value1" is selected and copied to output "TagOut_Value". If no errors occur during the execution of the instruction, the enable output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

MUX: Multiplex

Description

You can use the instruction "Multiplex" to copy the content of a selected input to output OUT. In its initial state the instruction box contains at least 2 inputs (IN0 and IN1). The number of selectable inputs of the instruction box can be expanded. The inputs are numbered automatically in the box. Numbering starts at IN0 and is incremented continuously with each new input. You can use the parameter K to determine the input whose content should be copied to output OUT. If the value of the parameter K is greater than the number of available inputs, the content of the parameter ELSE is copied to output OUT and enable output ENO is assigned the signal state "0".

The instruction "Multiplex" can only be executed if the tags at all inputs and at the output OUT are of the same data type. The exception here is the parameter K, which can only be specified as an integer.

The instruction is only executed if the signal state at the enable input EN is "1". If no error occurs during execution, output ENO also has the signal state "1".

The enable output ENO is reset if one of the following conditions applies:

- Enable Input EN has the signal state "0".
- The value of the parameter K is greater than the number of available inputs.
- Errors occurred during the execution of the instruction.

Parameters

The following table shows the parameters of the instruction "Multiplex":

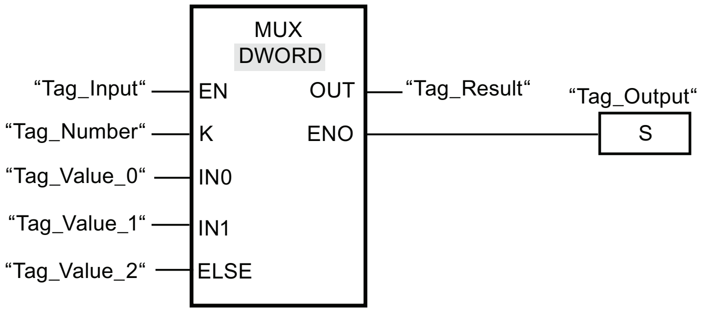
Parameters	Declaring	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
K	Input	UINT	I, Q, M, D, L or constant	Specifies the input whose content is to be copied.
IN0	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, CHAR	I, Q, M, D, L or constant	First input value
IN1	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, CHAR	I, Q, M, D, L or constant	Second input value
INn	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, CHAR	I, Q, M, D, L or constant	Optional input values
ELSE	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, CHAR	I, Q, M, D, L or constant	Specifies the value to be copied when K > n.
OUT	Output	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, CHAR	I, Q, M, D, L	Output to which the value is to be copied.

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
K	Tag_Number	1
IN0	Tag_ValueI_0	DW#16#00000000
IN1	Tag_Value_1	DW#16#3E4A7D
ELSE	Tag_Value_2	DW#16#FFFF0000
OUT	Tag_Result	DW#16#3E4A7D

If the operand "Tag_Input" has the signal state "1", the instruction "Multiplex" is executed. Depending on the value of the operand "Tag_Number", the value at input "Tag_Value_1" is copied and assigned to the operand at output "Tag_Result". If no errors occur during the execution of the instruction, the outputs ENO and "Tag_Output" are set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

DEMUX: Demultiplex

Description

The instruction "Demultiplex" copies the content of input IN to a selected output. In its initial state the instruction box contains at least 2 outputs (OUT0 and OUT1). The number of selectable outputs can be extended in the instruction box. The outputs are numbered automatically in the box. Numbering starts at OUT0 and is incremented continuously with each new input. You can use the parameter K to define the output to which the content of input IN will be copied. The other outputs will not be changed. If the value of the parameter K is greater than the number of available outputs, then the content of input IN in the parameter ELSE and the enable output ENO will be assigned to the signal state "0".

The instruction "Demultiplex" can only be executed if the tags at all input IN and at all outputs are of the same data type. The exception here is the parameter K, which can only be specified as an integer.

The instruction is only executed if the signal state at the enable input EN is "1". If no error occurs during execution, output ENO also has the signal state "1".

The enable output ENO is reset if one of the following conditions applies:

- Enable Input EN has the signal state "0".
- The value of the parameter K is greater than the number of available outputs.
- Errors occurred during the execution of the instruction.

Parameters

The following table shows the parameters of the instruction "Demultiplex":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
K	Input	UINT	I, Q, M, D, L or constant	Specifies the output to which the input value (IN) will be copied.
IN	Input	Bit strings, integers, floating-point numbers, CHAR, TIME	I, Q, M, D, L or constant	Input value
OUT0	Output	Bit strings, integers, floating-point numbers, CHAR, TIME	I, Q, M, D, L	First output
OUT1	Output	Bit strings, integers, floating-point numbers, CHAR, TIME	I, Q, M, D, L	Second output

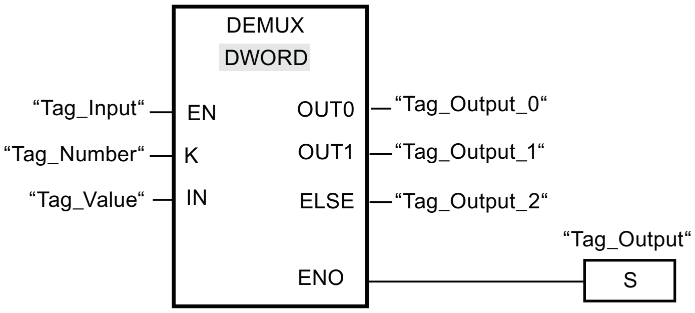
Parameters	Declaration	Data type	Memory area	Description
OUTn	Output	Bit strings, integers, floating-point numbers, CHAR, TIME	I, Q, M, D, L	Optional outputs
ELSE	Output	Bit strings, integers, floating-point numbers, CHAR, TIME	I, Q, M, D, L	Output to which the input value (IN) at K > n will be copied.

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on available data types, refer to "See also".

Example

The following example shows how the instruction works:



The following tables show how the instruction works using specific operand values:

Table 9- 24 Input values of the "Demultiplex" instruction before network execution

Parameters	Operand	Values	
K	Tag_Number	1	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#3E4A7D

Table 9- 25 Output values of the "Demultiplex" instruction after network execution

Parameters	Operand	Values	
OUT0	Tag_Output_0	Unchanged	Unchanged
OUT1	Tag_Output_1	DW#16#FFFFFFFF	Unchanged
ELSE	Tag_Output_2	Unchanged	DW#16#3E4A7D

If input "Tag_Input" has the signal state "1", the instruction "Demultiplex" is executed. Depending on the value of the operand "Tag_Number", the value at input "IN" is copied to the corresponding output.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

Shift and rotate

SHR: Shift right

Description

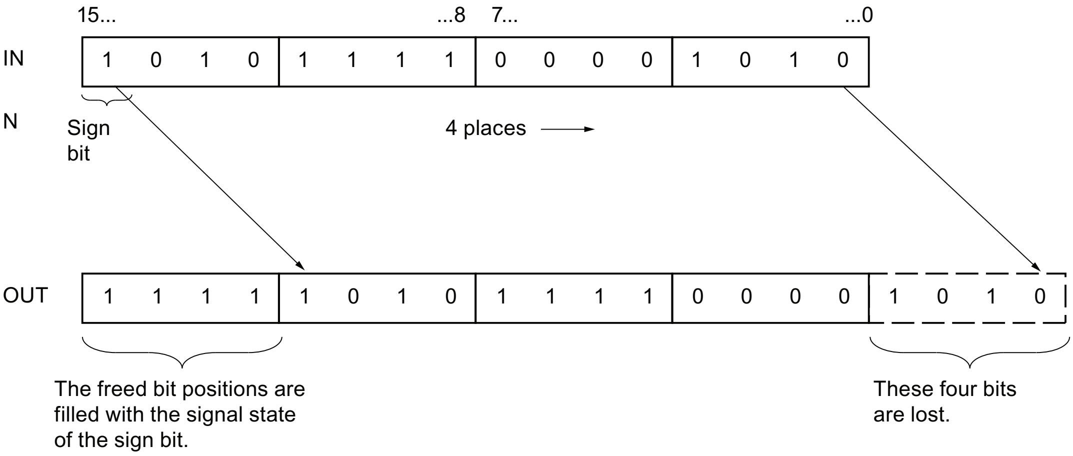
The instruction "Shift right" is used to move the content of the operand at input IN bit-by-bit to the right and query the result at output OUT. The input N is used to specify the number of bit positions by which the specified value should be moved.

If the value at the input N is "0", the value at input IN is copied unchanged to the operand at output OUT.

If the value at the input N is greater than the number of available bit positions, the operand value at input IN is shifted to the right by the available number of bit positions.

The freed bit positions in the left area of the operand are filled by zeroes when values without signs are shifted. If the specified value has a sign, the free bit positions are filled with the signal state of the sign bit.

The following figure show how the content of an integer data type operand is shifted four bit positions to the right:



The instruction "Shift right" is only executed if the signal state at the enable input EN is "1". In this case, the enable output ENO also has the signal state "1".

If the signal state at the enable input EN is "0", the signal state at the enable output ENO is also "0".

Parameters

The following table shows the parameters of the instruction "Shift right":

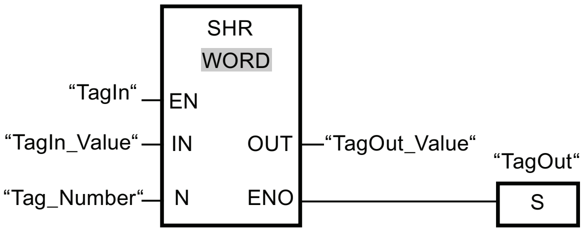
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	I, Q, M, D, L or constant	Value to be shifted.
N	Input	UINT	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted.
OUT	Output	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	3
OUT	TagOut_Value	0000 0111 1111 0101

If the operand "TagIn" has the signal state "1", the instruction "Shift right" is executed. The content of the operand "TagIn_Value" is shifted three bit positions to the right. The result is sent at output "TagOut_Value". If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

SHL: Shift left

Description

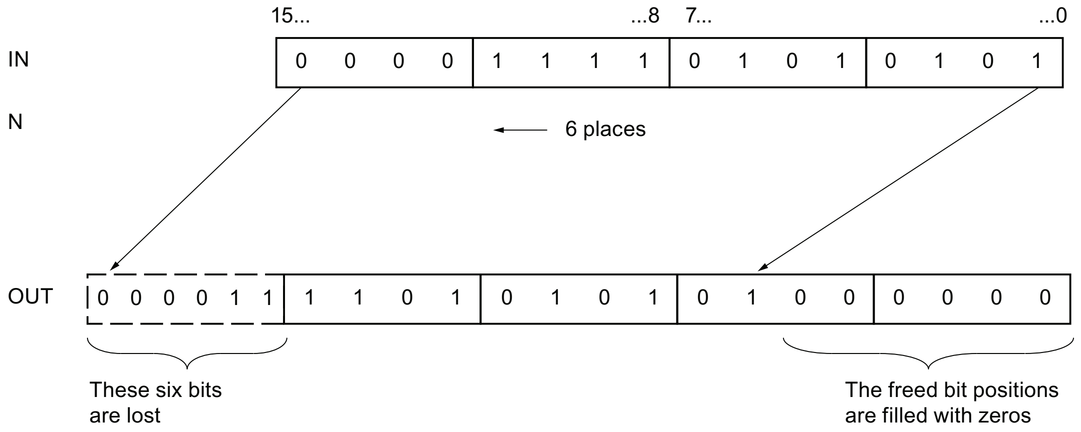
The instruction "Shift left" is used to move the content of the operand at input IN bit-by-bit to the left and query the result at output OUT. The input N is used to specify the number of bit positions by which the specified value should be moved.

If the value at the input N is "0", the value at input IN is copied unchanged to the operand at output OUT.

If the value at the input N is greater than the number of available bit positions, the operand value at input IN is shifted to the left by the available number of bit positions.

The bit positions in the right part of the operand freed by shifting are filled with zeros.

The following figure show how the content of a WORD data type operand is shifted six bit positions to the left:



The instruction "Shift left" is only executed if the signal state at the enable input EN is "1". In this case, the enable output ENO also has the signal state "1".

If the signal state at the enable input EN is "0", the signal state at the enable output ENO is also "0".

Parameters

The following table shows the parameters of the instruction "Shift left":

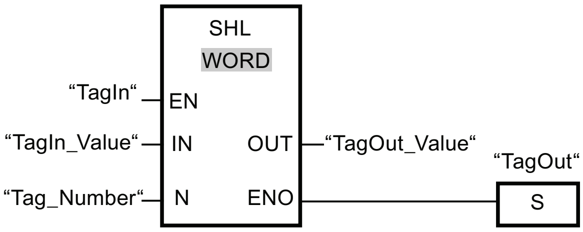
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	I, Q, M, D, L or constant	Value to be shifted.
N	Input	UINT	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted.
OUT	Output	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	4
OUT	TagOut_Value	1111 1010 1111 0000

If the operand "TagIn" has the signal state "1", the instruction "Shift left" is executed. The content of the operand "TagIn_Value" is shifted four bit positions to the left. The result is sent at output "TagOut_Value". If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

ROR: Rotate right

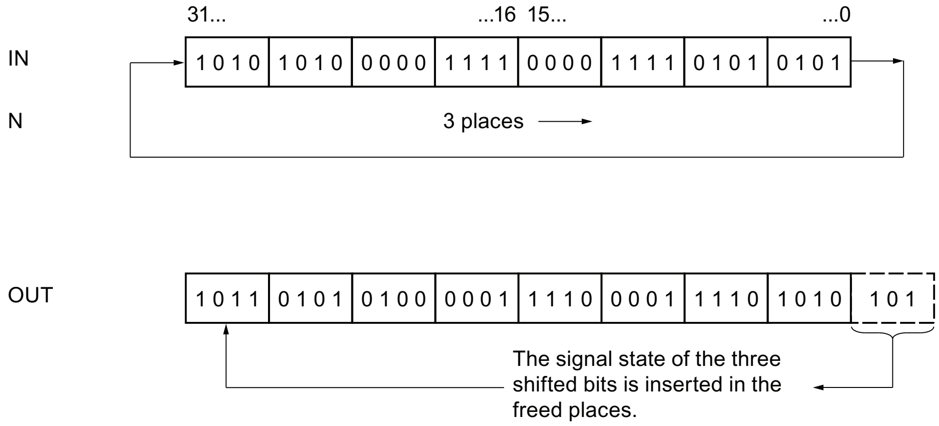
Description

The instruction "Rotate right" is used to rotate the content of the operand at input IN bit-by-bit to the right and query the result at output OUT. The input N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating on the left-hand side are filled true-to-position with the bit positions that are pushed out from the left-hand side.

If the value at the input N is "0", the value at input IN is copied unchanged to the operand at output OUT.

If the value at the parameter N is greater than the number of available bit positions, the operand value at input IN is nevertheless rotated by the specified number of bit positions.

The following figure show how the content of an DWORD data type operand is rotated three bit positions to the right:



The instruction "Rotate right" is only executed if the signal state at the enable input EN is "1". In this case, the enable output ENO also has the signal state "1".

If the signal state at the enable input EN is "0", the signal state at the enable output ENO is also "0".

Parameters

The following table shows the parameters of the instruction "Rotate right":

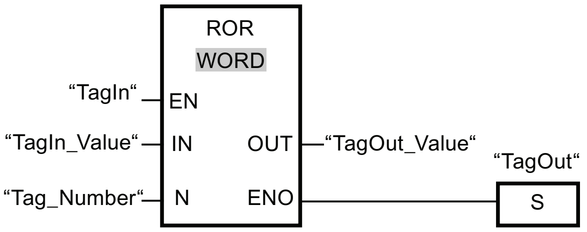
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings	I, Q, M, D, L or constant	Value to be rotated.
N	Input	UINT	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated.
OUT	Output	Bit strings	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0000 1111 1001 0101
N	Tag_Number	5
OUT	TagOut_Value	1010 1000 0111 1100

If the operand "TagIn" has the signal state "1", the instruction "Rotate right" is executed. The content of the operand "TagIn_Value" is rotated five bit positions to the right. The result is sent at output "TagOut_Value". If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

ROL: Rotate left

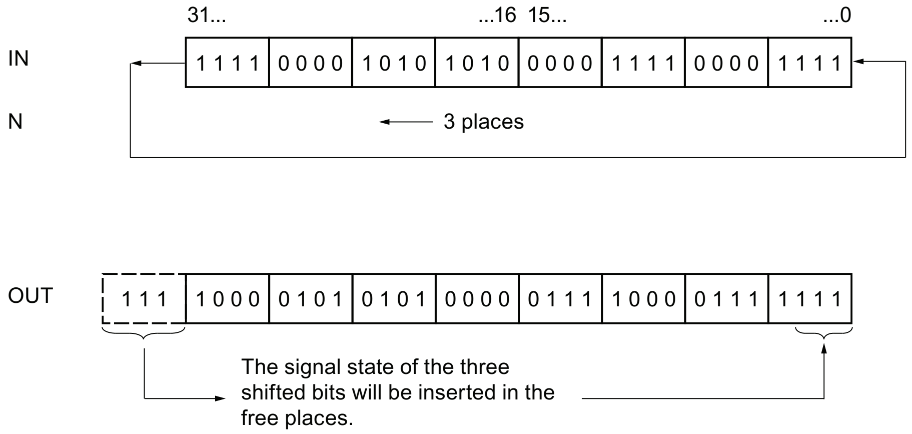
Description

The instruction "Rotate left" is used to rotate the content of the operand at input IN bit-by-bit to the left and query the result at output OUT. The input N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating on the right-hand side are filled true-to-position with the bit positions that are pushed out from the left-hand side.

If the value at the input N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the parameter N is greater than the number of available bit positions, the operand value at input IN is nevertheless rotated by the specified number of bit positions.

The following figure show how the content of an DWORD data type operand is rotated three bit positions to the left:



The instruction "Rotate left" is only executed if the signal state at the enable input EN is "1". In this case, the enable output ENO also has the signal state "1".

If the signal state at the enable input EN is "0", the signal state at the enable output ENO is also "0".

Parameters

The following table shows the parameters of the instruction "Rotate left":

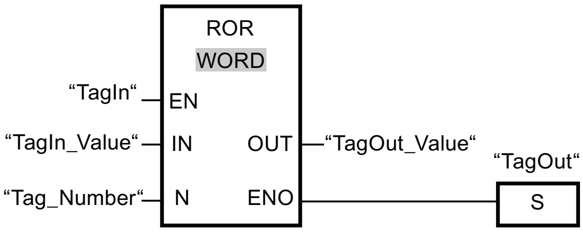
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings	I, Q, M, D, L or constant	Value to be rotated.
N	Input	UINT	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated.
OUT	Output	Bit strings	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	1010 1000 1111 0110
N	Tag_Number	5
OUT	TagOut_Value	0001 1110 1101 0101

If input "TagIn" has the signal state "1", the instruction "Rotate left" is executed. The content of the operand "TagIn_Value" is rotated five bit positions to the left. The result is sent at output "TagOut_Value". If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 741)

Basics of the EN/ENO mechanism (Page 819)

9.8.2.3 SCL

Timer operations

TP: Generate pulse

Description

The "Generate pulse" instruction sets the Q parameter for the time duration PT. The instruction is started when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. The Q parameter is set for the time PT, regardless of the subsequent changes in the input signal. Even when a new positive signal edge is detected, the signal state of the Q parameter is not affected as long as the PT time duration is running.

The current time value can be queried in the ET parameter. The time value starts at T#0s and ends when the value of the time duration PT is reached. When the time duration PT has been reached and the signal state of the IN parameter is "0", the ET parameter is reset.

Each call of the "Generate pulse" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER or TP that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the TP type in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. The query of the data is only equal to the call of the instruction until the next call to the instruction.

Syntax

Use the following syntax for the "Generate pulse" instruction:

- Data block of system data type IEC_Timer (global DB):

SCL

```
<IEC_Timer_DB>.TP(IN := <Operand>,
PT := <Operand>,
Q => <Operand>,
ET => <Operand>)
```

- Local tag:

SCL

```
#myLocal_timer(IN := <Operand>,
PT := <Operand>,
Q => <Operand>,
ET => <Operand>)
```

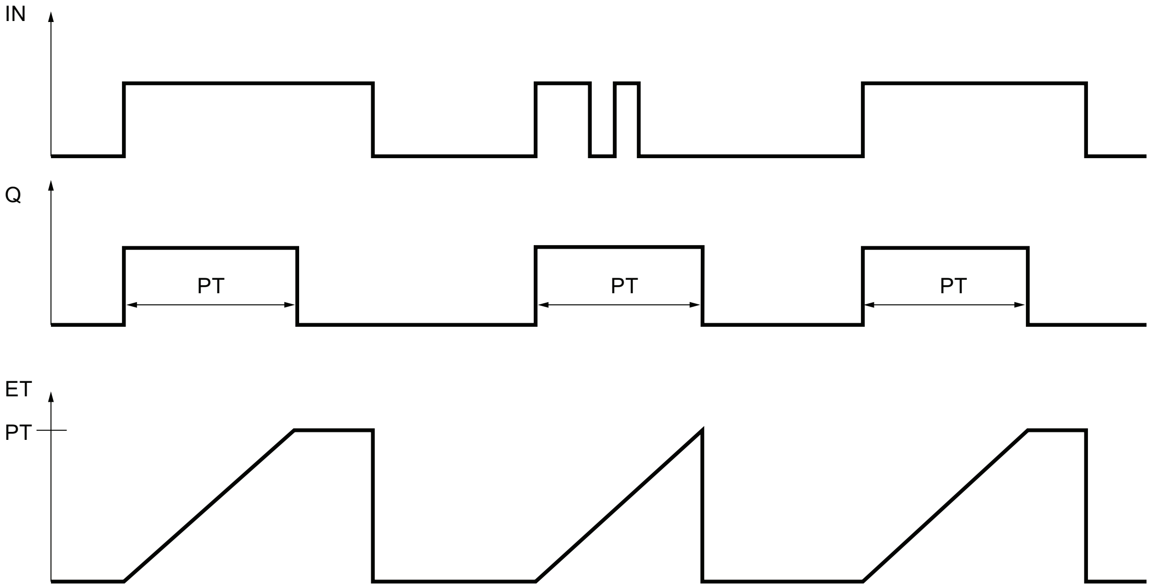
The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	Operand that is set for the PT duration.
ET	Output	TIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate pulse" instruction:



Example

The following example shows how the instruction works:

SCL

```
"TP_DB".TP(IN := "Tag_Start",
 PT := "Tag_PresetTime",
 Q => "Tag_Status",
 ET => "Tag_ElapsedTime");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time period programmed for the PT parameter is started and the "Tag_Status" operand is set to "1". The current time value is stored in the "Tag_ElapsedTime" operand.

See also

Overview of the valid data types (Page 741)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

TON: Generate on-delay

Description

The "Generate on-delay" instruction delays the setting of the Q parameter by the programmed time duration PT. The instruction is started when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the time PT has expired, the Q parameter returns signal state "1". The Q parameter remains set as long as the start input is still "1". If the signal state of the IN parameter changes from "1" to "0", the parameter Q is reset. The timer function is restarted when a new positive signal edge is detected at the IN parameter.

The current time value can be queried in the ET parameter. The time value starts at T#0s and ends when the value of the time duration PT is reached. The ET parameter is reset as soon as the signal state of the parameter IN changes to "0".

Each call of the "Generate on-delay" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER or TON that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the TON type in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. The query of the data is only equal to the call of the instruction until the next call to the instruction.

Syntax

Use the following syntax for the "Generate on-delay" instruction:

- Data block of system data type IEC_Timer (global DB):

```
SCL
<IEC_Timer_DB>.TON(IN := <Operand>,
PT := <Operand>,
Q => <Operand>,
ET => <Operand>)
```

- Local tag:

```
SCL
#myLocal_timer(IN := <Operand>,
PT := <Operand>,
Q => <Operand>,
ET => <Operand>)
```

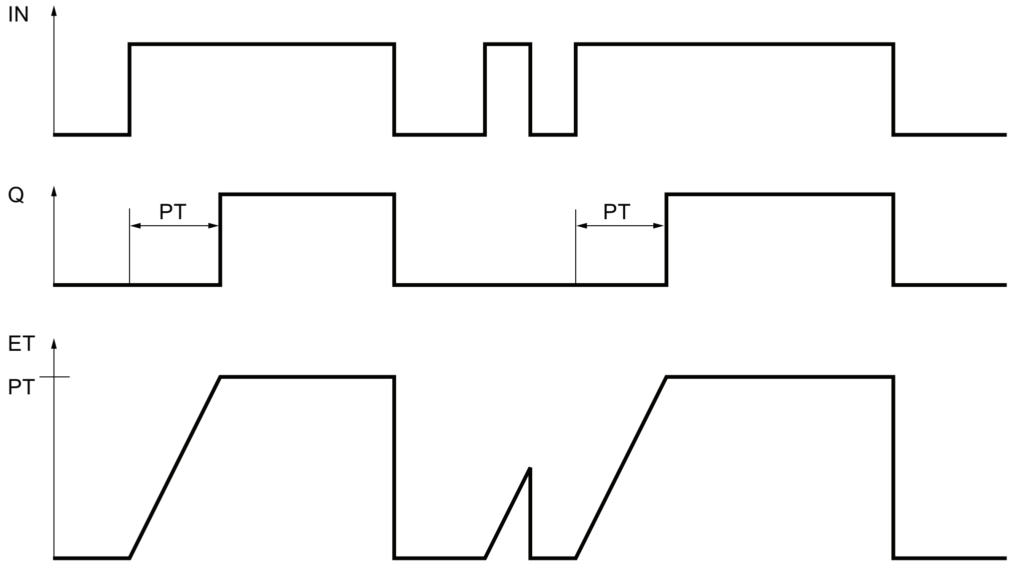
The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Duration of the on delay. The value of the PT parameter must be positive.
Q	Output	BOOL	Operand that is set when the timer PT expires.
ET	Output	TIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate on-delay" instruction:



Example

The following example shows how the instruction works:

SCL

```
"TON_DB".TON(IN := "Tag_Start",
 PT := "Tag_PresetTime",
 Q => "Tag_Status",
 ET => "Tag_ElapsedTime");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. At the end of the time, the "Tag_Start" operand is set to signal state "1" if the "Tag_Status" operand has signal state "1". At the end of the time, the "Tag_Status" operand is set to signal state "1" if the "Tag_Start" operand has signal state "1". The current count value is stored in the "Tag_ElapsedTime" operand.

See also

Overview of the valid data types (Page 741)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

TOF: Generate off-delay

Description

The "Generate off-delay" instruction delays the resetting of the Q parameter by the programmed time duration PT. The Q parameter is set when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). When the signal state of the IN parameter changes back to "0", the programmed time PT starts. The Q parameter remains set as long as the time duration PT is running. When the time PT expires, the Q parameter is reset. If the signal state of the IN parameter changes to "1" before the time duration PT expires, the timer is reset. The signal state of the Q parameter remains set to "1".

The current time value can be queried in the ET parameter. The time value starts at T#0s and ends when the value of the time duration PT is reached. When the time duration PT expires, the ET parameter remains set to the current value until the IN parameter changes back to "1". If the IN parameter changes to "1" before the time PT has expired, the ET parameter is reset to the value T#0s.

Each call of the "Generate off-delay" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER or TOF that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the TOF type in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. The query of the data is only equal to the call of the instruction until the next call to the instruction.

Syntax

The following syntax is used for the "Generate off-delay" instruction:

- Data block of system data type IEC_Timer (global DB):

SCL

```
<IEC_Timer_DB>.TOF(IN := <Operand>,
PT := <Operand>,
Q => <Operand>,
ET => <Operand>)
```

- Local tag:

SCL

```
#myLocal_timer(IN := <Operand>,
PT := <Operand>,
Q => <Operand>,
ET => <Operand>)
```

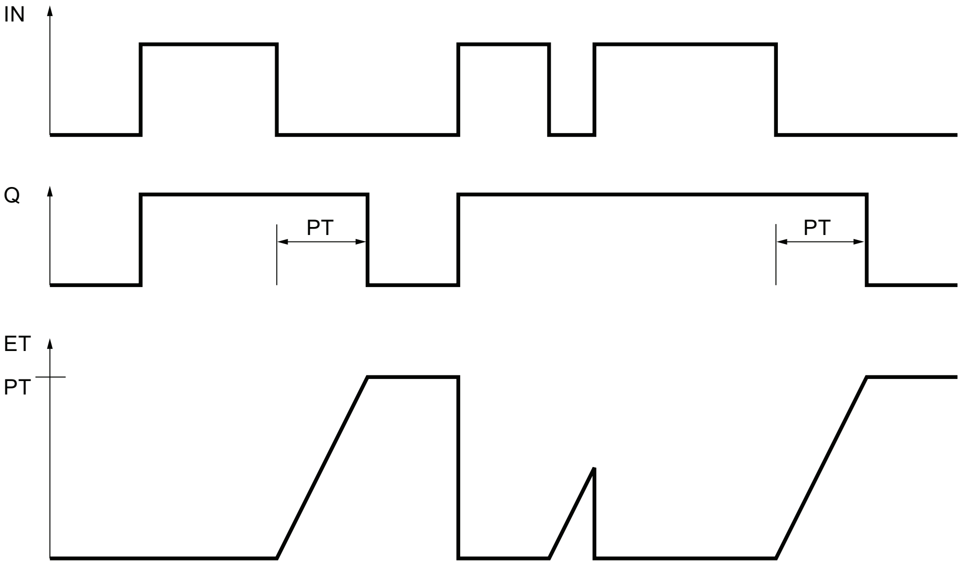
The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Duration of the off delay. The value of the PT parameter must be positive.
Q	Output	BOOL	Operand that is reset when the time PT expires.
ET	Output	TIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate off-delay" instruction:



Example

The following example shows how the instruction works:

SCL

```
"TOF_DB".TOF(IN := "Tag_Start",
 PT := "Tag_PresetTime",
 Q => "Tag_Status",
 ET => "Tag_ElapsedTime");
```

With a change in the signal state of the "Tag_Start" operand from "0" to "1", the "Tag_Status" operand is set. When the signal state of the "Tag_Start" operand changes from "1" to "0", the time programmed for the PT parameter is started. As long as the time is running, the "Tag_Status" operand remains set. When the time has expired, the "Tag_Status" operand is reset. The current count value is stored in the "Tag_ElapsedTime" operand.

See also

Overview of the valid data types (Page 741)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

TONR: Time accumulator

Description

The "Time accumulator" instruction accumulates time values within a time period set by parameter PT. When the signal state of the IN parameter changes to "1", the instruction executes and the time duration PT starts. While the time duration PT is running, the time values that are recorded when the IN parameter has signal state "1" are accumulated. The accumulated time is output in the ET parameter and can be queried there. When the time duration PT is reached, the Q parameter has signal state "1". The Q parameter remains set to "1", even when the signal state at the IN parameter changes to "0".

The R parameter resets the ET and Q parameters regardless of the signal state at the IN parameter.

Each call of the "Time accumulator" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER or TONR that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the TONR type in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. The query of the data is only equal to the call of the instruction until the next call to the instruction.

Syntax

Use the following syntax for the "Time accumulator" instruction:

- Data block of system data type IEC_Timer (global DB):

```
SCL
<IEC_Timer_DB>.TONR(IN := <Operand>,
 R := <Operand>,
PT := <Operand>,
Q => <Operand>,
ET => <Operand>)
```

- Local tag:

```
SCL
#myLocal_timer(IN := <Operand>,
 R := <Operand>,
PT := <Operand>,
Q => <Operand>,
ET => <Operand>)
```

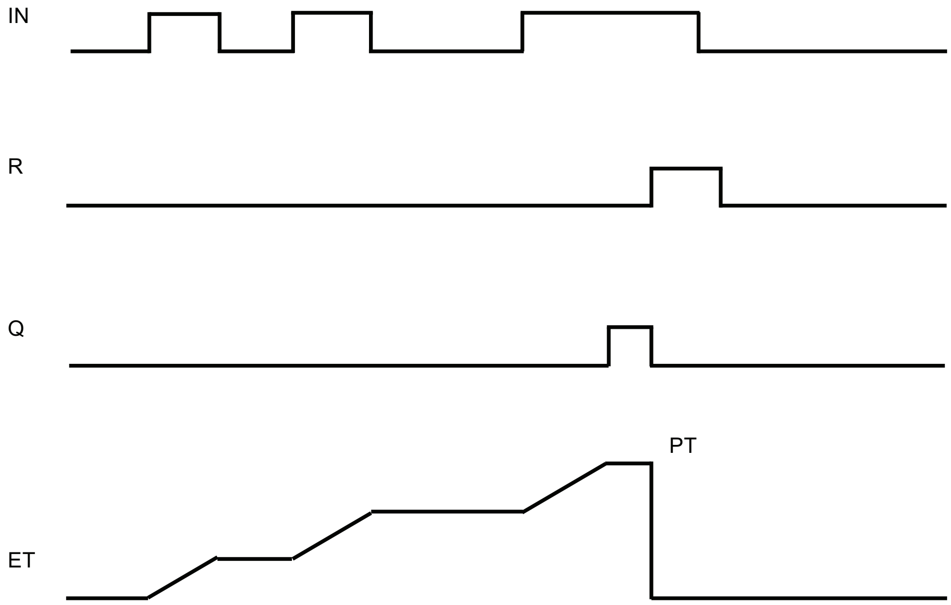
The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
IN	Input	BOOL	Start input
R	Input	BOOL	Reset of the ET and Q parameters
PT	Input	TIME	Maximum duration of time recording. The value of the PT parameter must be positive.
Q	Output	BOOL	Operand that remains set when the timer PT has expired.
ET	Output	TIME	Accumulated time

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Time accumulator" instruction:



Example

The following example shows how the instruction works:

SCL

```
"TONR_DB".TONR(IN := "Tag_Start",
 R := "Tag_Reset",
 PT := "Tag_PresetTime",
 Q => "Tag_Status",
 ET => "Tag_Time");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. While the timer is running, the timer values that are recorded at signal state "1" of the operand "Tag_Start" is accumulated. The accumulated times is stored in the "Tag_Time" operand. When the timer value displayed at the PT parameter is reached, the "Tag_Status" operand is set to the signal state "1". The current count value is stored in the "Tag_Time" operand.

See also

Overview of the valid data types (Page 741)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

Counter operations

CTU: Count up

Description

The "Count up" instruction increments the value at the CV parameter. When the signal state of the CU parameter changes from "0" to "1" (positive signal edge), the instruction is executed and the current count value of the CV parameter is incremented by one. When the instruction is executed for the first time the the current count of the CV parameter is set to zero. The count value is incremented each time a positive signal edge is detected, until it reaches the high limit of the data type specified for the CV parameter. When the high limit is reached, the signal state of the CU parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. The signal state of the Q parameter is determined by the PV parameter. When the current count value is greater than or equal to the value of the PV parameter, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0". You can also specify a constant for the PV parameter.

The value of the CV parameter is reset to zero when the signal state at the R parameter changes to "1". As long as the signal state of the R parameter is "1", the signal state of the CU parameter has no effect on the instruction.

Each call of the "Count up" instruction must be assigned to an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

- Data block of system data type IEC_counter (global DB):
  - IEC_SCOUNTER / IEC_USCOUNTER
  - IEC_COUNTER / IEC_UCOUNTER
  - IEC_DCOUNTER / IEC_UDCOUNTER
- Local tag:
  - CTU_SINT / CTU_USINT
  - CTU_INT / CTU_UINT
  - CTU_DINT / CTU_UDINT

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_COUNTER (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTU in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Count up" instruction:

- Data block of system data type IEC_counter (global DB):

```
SCL
<IEC_Counter_DB>.CTU(CU := <Operand>,
R := <Operand>,
PV := <Operand>,
Q => <Operand>,
CV => <Operand>)
```

- Local tag:

```
SCL
#myLocal_counter(CU := <Operand>,
R := <Operand>,
PV := <Operand>,
Q => <Operand>,
CV => <Operand>)
```

The following table shows the permitted data types for local counters:

Data type	Syntax	IEC counters
SINT	CTU_SINT	IEC_SCOUNTER / CTU_SINT
USINT	CTU_USINT	IEC_USCOUNTER / CTU_USINT
INT	CTU_INT	IEC_COUNTER / CTU_INT
UINT	CTU_UINT	IEC_UCOUNTER / CTU_UINT
DINT	CTU_DINT	IEC_DCOUNTER / CTU_DINT
UDINT	CTU_UDINT	IEC_UDCOUNTER / CTU_UDINT

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
CU	Input	BOOL	Count input
R	Input	BOOL	Reset input
PV	Input	Integers	Value at which the output Q is set
Q	Output	BOOL	Counter status
CV	Output	Integers	Current count value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"IEC_COUNTER_DB".CTU(CU := "Tag_Start",
 R := "Tag_Reset",
 PV := "Tag_PresetValue",
 Q => "Tag_Status",
 CV => "Tag_CounterValue");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the "Count up" instruction is executed and the current count value of the "Tag_CounterValue" operand is incremented by one. With each additional positive signal edge, the counter is incremented until the high limit value of the specified data type (32 767) is reached.

Output "Tag_Status" has the signal state "1" as long as the current counter is greater than or equal to the value of the operand "Tag_PresetValue". In all other cases, the "Tag_Status" output has signal state "0". The current count value is stored in the "Tag_CounterValue" operand.

See also

Overview of the valid data types (Page 741)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

CTD: Count down

Description

You can use the "Count down" instruction to decrement the value of the CV parameter. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the CV parameter is decremented by one. When the instruction is executed the first time, the counter value of the CV parameter will be set to the value of the PV parameter. Each time a positive signal edge is detected, the counter is decremented until it reaches the low limit value of the specified data type. When the low limit is reached, the signal state of the CD parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. If the current counter value is less than or equal to zero, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0". You can also specify a constant for the PV parameter.

The value of the CV parameter is set to the value of the PV parameter when the signal state of the LD parameter changes to "1". As long as the signal state of the LD parameter is "1", the signal state of the CD parameter has no effect on the instruction.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

- Data block of system data type IEC_counter (global DB):
  - IEC_SCOUNTER / IEC_USCOUNTER
  - IEC_COUNTER / IEC_UCOUNTER
  - IEC_DCOUNTER / IEC_UDCOUNTER
- Local tag:
  - CTU_SINT / CTU_USINT
  - CTU_INT / CTU_UINT
  - CTU_DINT / CTU_UDINT

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_COUNTER (for example, "MyIEC_COUNTER")
- Declaration as a local tag of CTD type in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Count down" instruction:

- Data block of system data type IEC_counter (global DB):

SCL

```
<IEC_Counter_DB> CTD(CD := <Operand>,
 LD := <Operand>,
 PV := <Operand>,
 Q => <Operand>,
 CV => <Operand>)
```

- Local tag:

SCL

```
#myLocal_counter(CD := <Operand>,
 LD := <Operand>,
 PV := <Operand>,
 Q => <Operand>,
 CV => <Operand>)
```

The following table shows the permitted data types for local counters:

Data type	Syntax	IEC counters
SINT	CTD_SINT	IEC_SCOUNTER / CTU_SINT
USINT	CTD_USINT	IEC_USCOUNTER / CTU_USINT
INT	CTD_INT	IEC_COUNTER / CTU_INT
UINT	CTD_UINT	IEC_UCOUNTER / CTU_UINT
DINT	CTD_DINT	IEC_DCOUNTER / CTU_DINT
UDINT	CTD_UDINT	IEC_UDCOUNTER / CTU_UDINT

The syntax of the instruction consists of the following parts:

Part/Parameter	Declaration	Data type	Description
CD	Input	BOOL	Count input
LD	Input	BOOL	Load input
PV	Input	Integers	Value at which the output Q is set
Q	Output	BOOL	Counter status
CV	Output	Integers	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"IEC_SCOUNTER_DB".CTD(CD := "Tag_Start",
 LD := "Tag_Load",
 PV := "Tag_PresetValue",
 Q => "Tag_Status",
 CV => "Tag_CounterValue");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the "Count down" instruction executes and the value of the "Tag_CounterValue" operand is decremented by one. With each additional positive signal edge, the counter value is decremented until it reaches the low limit of the specified data type (-128).

The operand "Tag_Status" has the signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_Status" output has signal state "0". The current counter value is stored in the "Tag_CounterValue" operand.

See also

Overview of the valid data types (Page 741)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

CTUD: Count up and down

Description

You can use the "Count up and down" instruction to increment and decrement the counter value of the CV parameter. When the signal state of the CU parameter changes from "0" to "1" (positive signal edge), the current counter value of the CV parameter is incremented by one. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the counter value of the CV parameter is decremented by one. If there is a positive signal edge at the CU and CD inputs in one program cycle, the current counter value of the CV parameter remains unchanged.

The counter value can be incremented until it reaches the high limit value of the data type specified for the CV parameter. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. When the low limit value of the specified data type is reached, the counter value is not decremented any further.

When the signal state of the LD parameter changes to "1", the counter value of the CV parameter is set to the value of the PV parameter. As long as the LD parameter has the signal state "1", the signal state of the CU and CD parameters has no effect on the instruction.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has signal state "1", a change in the the signal state of the CU, CD and LD parameters has no effect on the "Count up and down" instruction.

You can scan the current status of the up counter based on the value of the QU parameter. When the current counter value is greater than or equal to the value of the PV parameter, the QU parameter is set to signal state "1". In all other cases, the signal state of the QU parameter is "0". You can also specify a constant for the PV parameter.

You can scan the current status of the down counter based on the value of the QD parameter. If the current counter value is less than or equal to zero, the QD parameter is set to signal state "1". In all other cases, the signal state of the QD parameter is "0".

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

- Data block of system data type IEC_counter (global DB):
  - IEC_SCOUNTER / IEC_USCOUNTER
  - IEC_COUNTER / IEC_UCOUNTER
  - IEC_DCOUNTER / IEC_UDCOUNTER
- Local tag:
  - CTU_SINT / CTU_USINT
  - CTU_INT / CTU_UINT
  - CTU_DINT / CTU_UDINT

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_COUNTER (for example, "MyIEC_COUNTER")
- Declaration as a local tag of CTUD type in the "Input", "InOut" or "Static" section of a block (for example, #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Count up and down" instruction:

- Data block of system data type IEC_counter (global DB):

SCL

`<IEC_Counter_DB>.CTUD(CU:= <Operand>,`
`                      CD:= <Operand>,`
`                      R:= <Operand>,`
`                      LD:= <Operand>,`
`                      PV:= <Operand>,`
`                      QU=> <Operand>,`
`                      QD:= <Operand>,`
`                      CV=> <Operand>)`

- Local tag

SCL

`myLocal_counter(CU:= <Operand>,`
`                CD:= <Operand>,`
`                R:= <Operand>,`
`                LD:= <Operand>,`
`                PV:= <Operand>,`
`                QU=> <Operand>,`
`                QD:= <Operand>,`
`                CV=> <Operand>)`

The following table shows the permitted data types for local counters:

Data type	Syntax	IEC counters
SINT	CTUD_SINT	IEC_SCOUNTER / CTU_SINT
USINT	CTUD_USINT	IEC_USCOUNTER / CTU_USINT
INT	CTUD_INT	IEC_COUNTER / CTU_INT
UINT	CTUD_UINT	IEC_UCOUNTER / CTU_UINT
DINT	CTUD_DINT	IEC_DCOUNTER / CTU_DINT
UDINT	CTUD_UDINT	IEC_UDCOUNTER / CTU_UDINT

The syntax of the instruction consists of the following parts:

Part/Parameter	Declaration	Data type	Description
CU	Input	BOOL	Count up input
CD	Input	BOOL	Count down input
R	Input	BOOL	Reset input
LD	Input	BOOL	Load input
PV	Input	Integers	Value at which the output QU / QD is set.
QU	Output	BOOL	Status of the counter up
QD	Output	BOOL	Status of the counter down
CV	Output	Integers	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"IEC_COUNTER_DB".CTUD(CU := "Tag_Start1",
 CD := "Tag_Start2",
 LD := "Tag_Load",
 R := "Tag_Reset",
 PV := "Tag_PresetValue",
 QU => "Tag_CU_Status",
 QD => "Tag_CD_Status",
 CV => "Tag_CounterValue");
```

If the "Tag_Start1" operand has a positive signal edge in the signal state, the current counter value is incremented by one and stored in the "Tag_CounterValue" operand. If the "Tag_Start2" operand has a positive signal edge in the signal state, the counter value is decremented by one and is also stored in the "Tag_CounterValue" operand. The counter value is incremented on the positive signal edge of the CU parameter until it reaches the high limit of the specified data type (INT). If the CD parameter has a positive signal edge, the counter value is decremented until it reaches the low limit of the specified data type (INT).

The operand "Tag_CU_Status" has the signal state "1" as long as the current counter value is greater than or equal to the value of the operand "Tag_PresetValue". In all other cases, the "Tag_CU_Status" output has signal state "0".

The operand "Tag_CD_Status" has the signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_CD_Status" output has signal state "0".

See also

Overview of the valid data types (Page 741)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

Math functions

ABS: Form absolute value

Description

The "Form absolute value" instruction calculates the absolute value of an input value and saves the result in the specified operand.

Syntax

Use the following syntax for the instruction "Form absolute value":

SCL
`ABS(<Expression>)`

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
<Expression>	Input	SINT, INT, DINT, floating-point numbers	Input value
Function value		SINT, INT, DINT, floating-point numbers	Absolute value of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := ABS("Tag_Value");
"Tag_Result2" := ABS("Tag_Value1"*"Tag_Value2");
```

The absolute value of the input value is returned in the format of the input value as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	-2
Tag_Result1	2
Tag_Value1	4
Tag_Value2	-1
Tag_Result2	4

See also

Overview of the valid data types (Page 741)

Expressions (Page 975)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

MIN: Get minimum

Description

The "Get minimum" instruction compares the values of the available inputs and returns the lowest value as the result. The instruction is only executed if the tags of all inputs are of the same data type.

A minimum of two and a maximum of 32 inputs can be specified for the execution of the instruction.

Syntax

The following syntax is used for the "Get minimum" instruction:

SCL

```
MIN(IN1:= <Operand>,
IN2:= <Operand>,
INn := <Operand>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
IN1	Input	Integers, floating-point numbers, TIME, DATE, TOD	First input value
IN2	Input	Integers, floating-point numbers, TIME, DATE, TOD	Second input value
INn	Input	Integers, floating-point numbers, TIME, DATE, TOD	Optional input values (n=3 to 32)
Function value		Integers, floating-point numbers, TIME, DATE, TOD	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := MIN(IN1 := "Tag_Value1",
 IN2 := "Tag_Value2",
 IN3 := "Tag_Value3");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	12 222
IN2	Tag_Value2	14 444
IN3	Tag_Value3	13 333
Function value	Tag_Result	12 222

The instruction compares the values of the available inputs and copies the lowest value ("Tag_Value1") to operand "Tag_Result".

See also

Overview of the valid data types (Page 741)

Expressions (Page 975)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values of the available inputs and returns the highest value as the result. The instruction is only executed if the tags of all inputs are of the same data type.

A minimum of two and a maximum of 32 input values can be specified for the execution of the instruction.

Syntax

The following syntax is used for the "Get maximum" instruction:

SCL

```
MAX(IN1:= <Operand>,
IN2:= <Operand>,
INn := <Operand>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
IN1	Input	Integers, floating-point numbers, TIME, DATE, TOD	First input value
IN2	Input	Integers, floating-point numbers, TIME, DATE, TOD	Second input value
INn	Input	Integers, floating-point numbers, TIME, DATE, TOD	Optional input values (n=3 to 32)
Function value		Integers, floating-point numbers, TIME, DATE, TOD	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := MAX(IN1 := "Tag_Value1",
 IN2 := "Tag_Value2",
 IN3 := "Tag_Value3");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	12 222
IN2	Tag_Value2	14 444
IN3	Tag_Value3	13 333
Function value	Tag_Result	14 444

The instruction compares the values of the specified operands and copies the highest value ("Tag_Value2") to operand "Tag_Result".

See also

Overview of the valid data types (Page 741)

Expressions (Page 975)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

LIMIT: Set limit value

Description

The "Set limit value" instruction limits the value of the IN parameter to the values of the MN and MX parameters. The value of the MN parameter may not be greater than the value of the MX parameter.

If the value of the IN parameter fulfills the MN <= IN <= MX condition, it is sent as the result of the instruction. If the condition is not fulfilled and the input value IN is less than the low limit MN, the value of the MN parameter is sent as the result. If the high limit MX is exceeded, the value of the MX parameter is sent as a result.

The instruction is only executed if the operands of all parameters are of the same data type.

Syntax

The following syntax is used for the "Set limit value" instruction:

```
SCL
LIMIT(MN:= <Operand>,
IN := <Operand>,
MX := <Operand>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
MN	Input	Integers, floating-point numbers, TIME, DATE, TOD	Low limit
IN	Input	Integers, floating-point numbers, TIME, DATE, TOD	Input value
MX	Input	Integers, floating-point numbers, TIME, DATE, TOD	High limit
Function value		Integers, floating-point numbers, TIME, DATE, TOD	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
 SCL
"Tag_Result" := LIMIT(MN := "Tag_Minimum",
                      IN := "Tag_Value",
                      MX := "Tag_Maximum");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
MN	Tag_Minimum	12 000
IN	Tag_Value	8 000
MX	Tag_Maximum	16 000
Function value	Tag_Result	12 000

The value of operand "Tag_Value" is compared with the values of the "Tag_Minimum" and "Tag_Maximum" operands. Because the value of the "Tag_Value" operand is less than the lower limit, the value of the "Tag_Minimum" operand is copied to the "Tag_Result" operand.

See also

Overview of the valid data types (Page 741)

Expressions (Page 975)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

SQR: Form square

Description

The "Form square" instruction calculates the square of an input value and saves the result in the specified operand.

Syntax

Use the following syntax for the instruction "Form square":

SCL

```
SQR(<Expression>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
<Expression>	Input	Integers, floating-point numbers	Input value
Function value		Floating-point numbers	Square of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := SQR("Tag_Value");
"Tag_Result2" := SQR((SQR("Tag_Value1"))*"Tag_Value2);
```

The square of the input value is returned in the "Tag_Result" operand as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	2.5
Tag_Result1	6.25
Tag_Value1	6.0
Tag_Value2	2.0
Tag_Result2	5184.0

See also

Overview of the valid data types (Page 741)

Expressions (Page 975)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

SQRT: Form square root

Description

The "Form square root" instruction calculates the square root of an input value and saves the result in the specified operand. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number. If the input value is "-0", the result is also "-0".

Syntax

Use the following syntax for the instruction "Form square root":

SCL
SQRT(<Expression>)

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
<Expression>	Input	Integers, floating-point numbers	Input value
Function value		Floating-point numbers	Square root of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := SQRT("Tag_Value");
"Tag_Result2" := SQRT((SQR("Tag_Value1"))+"Tag_Value2");
```

The square root of the input value is returned in the "Tag_Result" operand as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	4.0
Tag_Result1	2.0
Tag_Value1	3.0
Tag_Value2	16.0
Tag_Result2	5.0

See also

Overview of the valid data types (Page 741)

Expressions (Page 975)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

LN: Form natural logarithm

Description

The "Form natural logarithm" instruction calculates the natural logarithm to base e (e = 2.718282e+00) from the input value. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, the instruction will return an invalid floating-point number.

Syntax

Use the following syntax for the instruction "Form natural logarithm":

```
SCL
LN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
<expression>	Input	Integers, floating-point numbers	Input value
Function value		Floating-point numbers	Natural logarithm of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := LN("Tag_Value");
"Tag_Result2" := LN("Tag_Value1"+"Tag_Value2");
```

The result of the instruction is returned in the "Tag_Result" operand as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	2.5
Tag_Result1	0.916
Tag_Value1	1.5
Tag_Value2	3.2
Tag_Result2	1.548

See also

Overview of the valid data types (Page 741)

Expressions (Page 975)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

EXP: Form exponential value

Description

The "Form exponential value" instruction calculates the power from the base e (e = 2.718282e+00) and the input value and saves the result in the specified operand.

Syntax

Use the following syntax for the instruction "Form exponential value":

```
SCL
EXP(<Expression>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
<Expression>	Input	Integers, floating-point numbers	Input value
Function value		Floating-point numbers	Exponential value of the input value IN

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := EXP("Tag_Value");
"Tag_Result2" := EXP("Tag_Value1"/"Tag_Value2");
```

The result of the instruction is returned in the "Tag_Result" operand as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	20.5
Tag_Result1	799902200
Tag_Value1	15.5
Tag_Value2	30.2
Tag_Result2	1.671

See also

Overview of the valid data types (Page 741)

Expressions (Page 975)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

SIN: Form sine value

Description

The "Form sine value" instruction calculates the sine of the input value. The input value must be given in the radian measure.

Syntax

Use the following syntax for the instruction "Form sine value":

SCL
`SIN(<Expression>)`

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
<Expression>	Input	Integers, floating-point numbers	Input value (size of an angle in the radian measure)
Function value		Floating-point numbers	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := SIN("Tag_Value");
```

The result of the instruction is returned in the "Tag_Result" operand as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+1.570796e+00 (π/2)
Tag_Result	1.0

See also

Overview of the valid data types (Page 741)

Expressions (Page 975)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

COS: Form cosine value

Description

The "Form cosine value" instruction calculates the cosine of the input value. The input value must be given in the radian measure.

Syntax

Use the following syntax for the "Form cosine value" instruction:

```
SCL
COS(<Expression>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
<Expression>	Input	Integers, floating-point numbers	Input value (size of an angle in the radian measure)
Function value		Floating-point numbers	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := COS("Tag_Value");
```

The result of the instruction is returned in the "Tag_Result" operand as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+1.570796e+00 (π/2)
Tag_Result	0

See also

Overview of the valid data types (Page 741)

Expressions (Page 975)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

TAN: Form tangent value

Description

The "Form tangent value" instruction calculates the tangent of the input value. The input value must be given in the radian measure.

Syntax

Use the following syntax for the instruction "Form tangent value":

```
SCL
TAN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
<Expression>	Input	Integers, floating-point numbers	Input value (size of an angle in the radian measure)
Function value		Floating-point numbers	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
 SCL
"Tag_Result" := TAN("Tag_Value");
```

The result of the instruction is returned in the "Tag_Result" operand as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+3.141593e+00 (π)
Tag_Result	0

See also

Overview of the valid data types (Page 741)

Expressions (Page 975)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

ASIN: Form arcsine value

Description

The "Form arcsine value" instruction uses the sine value to calculate the size of the angle that corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified as input values. The calculated angle size is given in the radian measure and can range in value from -π/2 to +π/2.

Syntax

Use the following syntax for the instruction "Form arcsine value":

SCL

```
ASIN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
<Expression>	Input	Integers, floating-point numbers	Sine value
Function value		Floating-point numbers	Size of angle in the radian measure

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ASIN("Tag_Value");
```

The result of the instruction is returned in the "Tag_Result" operand as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	1.0
Tag_Result	+1.570796e+00 (π/2)

See also

Overview of the valid data types (Page 741)

Expressions (Page 975)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

ACOS: Form arccosine value

Description

The "Form arccosine value" instruction uses a cosine value to calculate the size of the angle that corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified as input values. The calculated angle size is given in the radian measure and can range in value from 0 to +π.

Syntax

Use the following syntax for the instruction "Form arccosine value":

SCL

```
ACOS(<Expression>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
<Expression>	Input	Integers, floating-point numbers	Cosine value
Function value		Floating-point numbers	Size of angle in the radian measure

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := ACOS("Tag_Value");
```

The result of the instruction is returned in the "Tag_Result" operand as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	0
Tag_Result	+1.570796e+00 (π/2)

See also

Overview of the valid data types (Page 741)

Expressions (Page 975)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

ATAN: Form arctangent value

Description

The "Form arctangent value" instruction uses the tangent value to calculate the size of the angle that corresponds to this value. Only valid floating-point numbers may be specified as input values. The calculated angle size is given in the radian measure and can range in value from -π/2 to +π/2.

Syntax

Use the following syntax for the instruction "Form arctangent value":

SCL

```
ATAN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
<Expression>	Input	Integers, floating-point numbers	Tangent value
Function value		Floating-point numbers	Size of angle in the radian measure

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ATAN("Tag_Value");
```

The result of the instruction is returned in the "Tag_Result" operand as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	1.0
Tag_Result	+0.785398e+00 (π/4)

See also

Overview of the valid data types (Page 741)

Expressions (Page 975)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

Move operations

MOVE_BLK: Move block

Description

The "Move block" instruction copies the content of a memory area (source area) to a different memory area (target area). The number of elements to be copied to the destination area is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the first element in the source area. The copy operation takes place in the direction of ascending addresses.

Syntax

The following syntax is used for the "Move block" instruction:

```
SCL
MOVE_BLK(IN := <operand>,
 COUNT := <operand>,
 OUT => <Operand>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
IN	Input	BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	The first element of the source area to be copied.
COUNT	Input	UINT	Number of elements to be copied from the source area to the destination area.
OUT	Output	BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	The first element of the destination area to which the content of the source area is copied.
Function value		BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := MOVE_BLK(IN := #a_array[2],
 COUNT := "Tag_Count",
OUT => #b_array[1]);
```

The result of the instruction is returned as a function value.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	a_array[2]	The operand "a_array" is an ARRAY data type and consists of five elements of the INT data type.
COUNT	Tag_Count	3
OUT	b_array[1]	The operand "b_array" is an ARRAY data type and consists of six elements of the INT data type.

The instruction selects three INT elements from the tag "a_array" (a_array[2..4]) and moves their content to the tag "b_array" (b_array[1..3]).

See also

Overview of the valid data types (Page 741)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

UMOVE_BLK: Move block uninterruptible

Description

The "Move block uninterruptible" instruction copies the content of a memory area (source area) to a different memory area (target area) without interruption. The number of elements to be copied to the destination area is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the first element in the source area. The copy operation takes place in the direction of ascending addresses.

The copy operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the "Move block uninterruptible" instruction.

Syntax

The following syntax is used for the "Move block uninterruptible" instruction:

SCL

```
UMOVE_BLK(IN := <operand>,
 COUNT := <operand>,
 OUT => <operand>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
IN	Input	BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	The first element of the source area to be copied.
COUNT	Input	UINT	Number of elements to be copied from the source area to the destination area.
OUT	Output	BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	The first element of the destination area to which the content of the source area is copied.
Function value		BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := UMOVE_BLK(IN := #a_array[2],
 COUNT := "Tag_Count",
 OUT => #b_array[1]);
```

The result of the instruction is returned as a function value.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	a_array[2]	The operand "a_array" is an ARRAY data type and consists of five elements of the INT data type.
COUNT	Tag_Count	3
OUT	b_array[1]	The operand "b_array" is an ARRAY data type and consists of six elements of the INT data type.

The instruction selects three INT elements from the "a_array"(a_array[2 ... 4]) tags and copies their content to the "b_array" (b_array[1 ... 3]) output tag. The copy operation cannot be interrupted by other operating system activities.

See also

Overview of the valid data types (Page 741)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

FILL_BLK: Fill block

Description

The "Fill block" instruction fills a memory area (target area) with the content of a different memory area (source area). The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the source area is selected and moved to the destination area as often as specified by the value of the COUNT parameter.

Syntax

The following syntax is used for the "Fill block" instruction:

```
SCL
FILL_BLK(IN := <operand>,
 COUNT := <operand>,
 OUT => <Operand>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
IN	Input	BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	Element used to fill the destination area.
COUNT	Input	UINT	Number of repeated copy operations
OUT	Output	BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	Address in destination area where filling begins.
Function value		BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := FILL_BLK(IN := #a_array[2],
 COUNT := "Tag_Count",
 OUT => #b_array[1]);
```

The result of the instruction is returned as a function value.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	a_array[2]	The operand "a_array" is an ARRAY data type and consists of four elements of the WORD data type (ARRAY[1 to 4] of WORD).
COUNT	Tag_Count	3
OUT	b_array[1]	The operand "b_array" is an ARRAY data type and consists of five elements of the WORD data type (ARRAY[1 to 5] of WORD).

The instruction copies the second element (a_array[2]) of the tag "a_array" three times to the output tag "b_array" (b_array[1..3]).

See also

Overview of the valid data types (Page 741)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

UFILL_BLK: Fill block uninterruptible

Description

The "Fill block uninterruptible" instruction fills a memory area (target area) with the content of a different memory area (source area) without interruption. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at input IN is selected and copied to the destination area as often as specified by the value of the COUNT parameter.

The copy operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the "Fill block uninterruptible" instruction.

Syntax

The following syntax is used for the "Fill block uninterruptible" instruction:

```
SCL
UFILL_BLK(IN := <operand>,
 COUNT := <operand>,
 OUT => <Operand>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
IN	Input	BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	Element used to fill the destination area.
COUNT	Input	UINT	Number of repeated copy operations
OUT	Output	BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	Address in destination area where filling begins.
Function value		BOOL, bit strings, integers, floating-point numbers, timers, DATE and CHAR as components of an ARRAY structure	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := UFILL_BLK(IN := #a_array[2],
 COUNT := "Tag_Count",
 OUT => #b_array[1]);
```

The result of the instruction is returned as a function value.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	a_array[2]	The operand "a_array" is an ARRAY data type and consists of four elements of the WORD data type (ARRAY[1..4] of WORD).
COUNT	Tag_Count	3
OUT	b_array[1]	The operand "b_array" is an ARRAY data type and consists of five elements of the WORD data type (ARRAY[1..5] of WORD).

The instruction copies the second element (a_array[2]) of the tag "a_array" three times to the output tag "b_array" (b_array[1..3]). The copy operation cannot be interrupted by other operating system activities.

See also

Overview of the valid data types (Page 741)

Entering SCL instructions (Page 991)

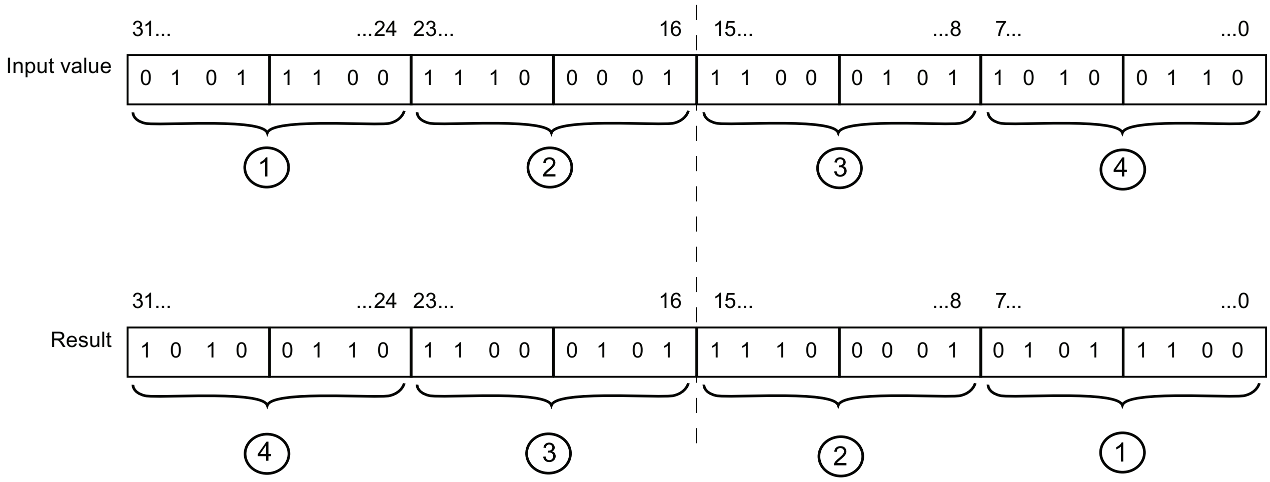
Editing SCL instructions (Page 1005)

SWAP: Swap

Description

The "Swap" instruction changes the arrangement of the bytes of an input value and saves the result in the specified operand.

The following figure shows how the bytes of an operand of the DWORD data type are swapped using the "Swap" instruction:



Syntax

The following syntax is used for the "Swap" instruction:

SCL
`SWAP(<Expression>)`

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
<Expression>	Input	WORD, DWORD	Input value
Funktionswert		WORD, DWORD	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL
`"Tag_Result" := SWAP("Tag_Value");`

The result of the instruction is returned as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	0000 1111 0101 0101
Tag_Result	0101 0101 1111 0000

See also

Overview of the valid data types (Page 741)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

Conversion operations

CONVERT: Convert value

Description

Use the "Convert value" instruction to program explicit conversions. You determine the data types to be converted in a dialogue box with opens automatically when you insert the instruction. When executed, the instruction reads the source value and converts it into the specified target value.

For information on possible conversions, refer to the "Explicit conversion (Page 799)" section.

Syntax

The "Convert value" instruction uses the syntax of the explicit conversion functions. For information on this function, refer to the "Explicit conversion functions (Page 1617)" section.

Example

The following example shows how the instruction works:

SCL

```
"Tag_INT" := REAL_TO_INT("Tag_REAL");
```

The following table shows how the instruction works using specific operand values:

Operand	Data type	Value
Tag_REAL	REAL	20.56
Tag_INT	INT	21

With conversion, the value of the "Tag_REAL" operand is rounded to the nearest integer and saved in the "Tag_INT" " operand.

See also

Overview of the valid data types (Page 741)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

Explicit conversion functions

Description

You can program explicit data type conversions using the conversion functions. You determine the data types to be converted by the syntax of the individual functions according to the following schema:

<Data type_of_the_source_value>_TO_<Data_type_of_the_target value>

The following table shows the available conversion functions:

Table 9- 26 Conversions of BYTE

Function	Explanation
BYTE_TO_WORD	The bit sample of the source value is transferred right-justified without any alterations into the target data type.
BYTE_TO_DWORD	
BYTE_TO_SINT	
BYTE_TO_USINT	
BYTE_TO_INT	
BYTE_TO_UINT	
BYTE_TO_DINT	
BYTE_TO_UDINT	

Table 9- 27 Conversions of WORD

Function	Explanation
WORD_TO_BYTE	The bit sample of the source value is transferred right-justified without any alterations into the target data type. If the permitted value range of the target data type is exceeded, the ENO enable output is set to "0". In this case, the result of the conversion is invalid.
WORD _TO_DWORD	
WORD _TO_SINT	
WORD _TO_USINT	
WORD _TO_INT	
WORD _TO_UINT	
WORD _TO_DINT	
WORD _TO_UDINT	

Table 9- 28 Conversions of DWORD

Function	Explanation
DWORD_TO_BYTE	The bit sample of the source value is transferred right-justified without any alterations into the target data type. If the permitted value range of the target data type is exceeded, the ENO enable output is set to "0". In this case, the result of the conversion is invalid.
DWORD_TO_WORD	
DWORD_TO_SINT	
DWORD_TO_USINT	
DWORD_TO_INT	
DWORD_TO_UINT	
DWORD_TO_DINT	
DWORD_TO_UDINT	
DWORD_TO_REAL	

Table 9- 29 Conversions of SINT

Function	Explanation
SINT_TO_BYTE	The bit sample of the source value is transferred right-justified without any alterations into the target data type. If a negative value is converted into an unsigned target data type, the ENO enable output is set to "0".
SINT_TO_WORD	
SINT_TO_DWORD	
SINT_TO_USINT	
SINT_TO_INT	
SINT_TO_UINT	
SINT_TO_DINT	
SINT_TO_UDINT	
SINT_TO_REAL	The value is converted into the format of the target data type. The value "-1" is changed to the value "-1.0", for example, with the "Convert value" instruction.
SINT_TO_LREAL	
SINT_TO_CHAR	The bit pattern of the source value is transferred without any changes into the target data type. With the conversion of negative values, the enable output ENO is set to "0".
SINT_TO_STRING	The value is converted into a character string. The character string is shown preceded by a sign. If the length of the character string is exceeded, the enable output "ENO" is set to "0".

Table 9- 30 Conversions of USINT

Function	Explanation
USINT_TO_BYTE	The bit sample of the source value is transferred right-justified without any alterations into the target data type.
USINT_TO_WORD	
USINT_TO_DWORD	
USINT_TO_SINT	The bit pattern of the source value is transferred without any changes into the target data type. If the sign bit is overwritten during conversion, the enable output ENO is set to "0".
USINT_TO_INT	The bit sample of the source value is transferred right-justified without any alterations into the target data type.
USINT_TO_UINT	
USINT_TO_DINT	
USINT_TO_UDINT	
USINT_TO_REAL	The value is converted into the format of the target data type. The value "-1" is changed to the value "-1.0", for example, with the "Convert value" instruction.
USINT_TO_LREAL	
USINT_TO_CHAR	The bit pattern of the source value is transferred without any changes into the target data type.
USINT_TO_STRING	The value is converted into a character string. If the length of the character string is exceeded, the enable output "ENO" is set to "0".

Table 9- 31 Conversions of INT

Function	Explanation
INT_TO_BYTE	The bit sample of the source value is transferred right-justified without any alterations into the target data type. If a negative value is converted into an unsigned target data type or an overflow occurs, the ENO enable output is set to "0".
INT_TO_WORD	
INT_TO_DWORD	
INT_TO_SINT	
INT_TO_USINT	
INT_TO_UINT	
INT_TO_DINT	
INT_TO_UDINT	
INT_TO_REAL	The value is converted into the format of the target data type. The value "-1" is changed to the value "-1.0", for example, with the "Convert value" instruction.
INT_TO_LREAL	
INT_TO_CHAR	The bit pattern of the source value is transferred without any changes into the target data type. With the conversion of negative values or with an overflow, the enable output ENO is set to "0".
INT_TO_STRING	The value is converted into a character string. The character string is shown preceded by a sign. If the length of the character string is exceeded, the enable output "ENO" is set to "0".

Table 9- 32 Conversions of UINT

Function	Explanation
UINT_TO_BYTE	The bit sample of the source value is transferred right-justified without any alterations into the target data type. With an overflow, the enable output ENO is set to "0".
UINT_TO_WORD	
UINT_TO_DWORD	
UINT_TO_SINT	
UINT_TO_USINT	
UINT_TO_INT	The bit pattern of the source value is transferred without any changes into the target data type. If the sign bit is overwritten during conversion, the enable output ENO is set to "0".
UINT_TO_DINT	The bit sample of the source value is transferred right-justified without any alterations into the target data type.
UINT_TO_UDINT	
UINT_TO_REAL	The value is converted into the format of the target data type. The value "-1" is changed to the value "-1.0", for example, with the "Convert value" instruction.
UINT_TO_LREAL	
UINT_TO_DATE	The bit pattern of the source value is transferred without any changes into the target data type.
UINT_TO_CHAR	The bit pattern of the source value is transferred without any changes into the target data type. With an overflow, the enable output ENO is set to "0".
UINT_TO_STRING	The value is converted into a character string. If the length of the character string is exceeded, the enable output "ENO" is set to "0".

Table 9- 33 Conversions of DINT

Function	Explanation
DINT_TO_BYTE	The bit sample of the source value is transferred right-justified without any alterations into the target data type. If a negative value is converted into an unsigned target data type or an overflow occurs, the ENO enable output is set to "0".
DINT_TO_WORD	
DINT_TO_DWORD	
DINT_TO_SINT	
DINT_TO_USINT	
DINT_TO_INT	
DINT_TO_UINT	
DINT_TO_UDINT	
DINT_TO_REAL	The value is converted into the format of the target data type. The value "-1" is changed to the value "-1.0", for example, with the "Convert value" instruction.
DINT_TO_LREAL	
DINT_TO_TIME	The bit pattern of the source value is transferred without any changes into the target data type.
DINT_TO_CHAR	The bit pattern of the source value is transferred without any changes into the target data type. With the conversion of negative values or with an overflow, the enable output ENO is set to "0".
DINT_TO_STRING	The value is converted into a character string. The character string is shown preceded by a sign. If the length of the character string is exceeded, the enable output "ENO" is set to "0".

Table 9- 34 Conversions of UDINT

Function	Explanation
UDINT_TO_BYTE	The bit pattern of the source value is transferred without any changes into the target data type. With an overflow, the enable output ENO is set to "0".
UDINT_TO_WORD	
UDINT_TO_DWORD	
UDINT_TO_SINT	
UDINT_TO_USINT	
UDINT_TO_INT	
UDINT_TO_UINT	
UDINT_TO_DINT	The bit pattern of the source value is transferred without any changes into the target data type. If the sign bit is overwritten during conversion, the enable output ENO is set to "0".
UDINT_TO_REAL	The value is converted into the format of the target data type. The value "-1" is changed to the value "-1.0", for example, with the "Convert value" instruction.
UDINT_TO_LREAL	
UDINT_TO_TOD	The bit pattern of the source value is transferred without any changes into the target data type.
UDINT_TO_CHAR	The bit pattern of the source value is transferred without any changes into the target data type. With an overflow, the enable output ENO is set to "0".
UDINT_TO_STRING	The value is converted into a character string. If the length of the character string is exceeded, the enable output "ENO" is set to "0".

Table 9- 35 Conversions of REAL

Function	Explanation
REAL_TO_DWORD	The bit pattern of the source value is transferred without any changes into the target data type.
REAL_TO_SINT	The value is converted into a target data type. The result of the conversion depends on the instruction used. If the permitted value range of the target data type is exceeded with the conversion or the value to be converted is an invalid floating-point number, the enable output ENO is set to "0".
REAL_TO_USINT	
REAL_TO_INT	
REAL_TO_UINT	
REAL_TO_DINT	
REAL_TO_UDINT	
REAL_TO_LREAL	The value is converted into a target data type.
REAL_TO_STRING	The value is converted into a character string. If the length of the character string is exceeded with the conversion or the value to be converted is an invalid floating-point number, the enable output ENO is set to "0".

Table 9- 36 Conversions of LREAL

Function	Explanation
LREAL_TO_SINT	The value is converted into a target data type. The result of the conversion depends on the instruction used. If the permitted value range is exceeded with the conversion or the value to be converted is an invalid floating-point number, the enable output ENO is set to "0".
LREAL_TO_USINT	
LREAL_TO_INT	
LREAL_TO_UINT	
LREAL_TO_DINT	
LREAL_TO_UDINT	
LREAL_TO_REAL	The value is converted into a target data type. If the permitted value range is exceeded with the conversion or the value to be converted is an invalid floating-point number, the enable output ENO is set to "0".
LREAL_TO_STRING	The value is converted into a character string. If the length of the character string is exceeded with the conversion or the value to be converted is an invalid floating-point number, the enable output ENO is set to "0".

Table 9- 37 Conversions of TIME

Function	Explanation
TIME_TO_DINT	The bit pattern of the source value is transferred without any changes into the target data type. The result of the conversion shows the duration in milliseconds.

Table 9- 38 Conversions of DATE

Function	Explanation
DATE_TO_UINT	The bit pattern of the source value is transferred without any changes into the target data type. The result of the conversion corresponds with the number of days since 01.01.1990.

Table 9- 39 Conversions of TOD

Function	Explanation
TOD_TO_UDINT	The bit pattern of the source value is transferred without any changes into the target data type. The result of the conversion corresponds with the number of milliseconds since the start of the day (00:00).

Table 9- 40 Conversions of DTL

Function	Explanation
DTL_TO_DATE	With the conversion, the information regarding the date is extracted from the DTL format and written into the target data type.
DTL_TO_TOD	With the conversion, the information regarding the time is extracted from the DTL format and written into the target data type.

Table 9- 41 Conversions of CHAR

Function	Explanation
CHAR_TO_SINT	The bit sample of the source value is transferred right-justified without any alterations into the target data type.
CHAR_TO_USINT	
CHAR_TO_INT	
CHAR_TO_UINT	
CHAR_TO_DINT	
CHAR_TO_UDINT	
CHAR_TO_STRING	The value is converted into the first character of the character string (STRING). If the length of the character string is not defined, the length "1" is set after conversion. If the length of the character string is defined, this length is the same after conversion.

Table 9- 42 Conversions of STRING

Function	Explanation
STRING_TO_SINT	The conversion starts with the first character of the string and ends with the end of the string or the first character that is invalid. The following characters are permitted for conversion: • Digit • Sign • Dot The first character of the string may be a sign (+, -) or a number. Leading spaces are ignored. The dot is used as separation for the conversion of floating-point numbers. If the structure of the string is invalid for the conversion or if an overflow occurs, the enable output ENO is set to "0".
STRING_TO_USINT	
STRING_TO_INT	
STRING_TO_UINT	
STRING_TO_DINT	
STRING_TO_UDINT	
STRING_TO_REAL	
STRING_TO_LREAL	
STRING_TO_CHAR	The first character of the string is transferred to the destination data type. If the string is empty, the value "0" is written in the destination data type.

Syntax

Use the following syntax for the conversion instructions:

```
<Target_value> := <Conversion_function>(<Source_value>);
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
<Source_value>	Input, constant	Bit sequences, integers, floating-point numbers, timers, date and time, character sequences	Value to be converted.
<Conversion_function>	-	-	Function that specifies the data type to be converted .
<Target_value>	Output	Bit sequences, integers, floating-point numbers, timers, date and time, character sequences	Result of the conversion

For additional information on valid data types, refer to "See also".

Example

The following example shows how the conversion function works:

```
SCL
"Tag_Word" := BYTE_TO_WORD("Tag_Byte");
```

The following table shows how the instruction works using specific operand values:

Operand	Data type	Value
Tag_Byte	BYTE	11010001
Tag_Word	WORD	0000000011010001

See also

Overview of the valid data types (Page 741)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

ROUND: Round numerical value

Description

The "Round numerical value" instruction rounds off the value at the IN input to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts it into an integer of the data type DINT. If the input value is exactly between an even and odd number, the even number is selected.

Syntax

Use the following syntax for the "Round numerical value" instruction:

```
SCL
ROUND(<Expression>)
```

The syntax of the instruction consists of the following parts:

Part/Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value to be rounded.
Function value		DINT	Result of the rounding

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ROUND("Tag_Value");
```

The result of the instruction is returned in the "Tag_Result" operand as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	0.50000000	-0.50000000
Tag_Result	0	0

See also

Overview of the valid data types (Page 741)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

TRUNC: Truncate numerical value

Description

The "Truncate numerical value" instruction forms an integer from an input value without rounding. The instruction selects only the integer part of the input value and returns this part without decimal places as the function value.

Syntax

Use the following syntax for the "Truncate numerical value" instruction:

SCL

```
TRUNC(<Expression>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value
Function value		DINT	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := TRUNC("Tag_Value1");
"Tag_Result2" := TRUNC("Tag_Value2"+"Tag_Value3");
```

The result of the instruction is returned in the "Tag_Result" operand as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value1	-1.5
Tag_Result1	-1
Tag_Value2	2.1
Tag_Value3	3.2
Tag_Result2	5

See also

Overview of the valid data types (Page 741)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

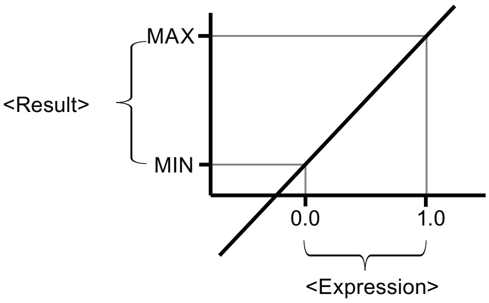
Editing SCL instructions (Page 1005)

SCALE_X: Scale

Description

The "Scale" instruction scales a floating-point number by mapping it to a specific range of values. You specify the value range with the MIN and MAX parameters. The result of the scaling is an integer.

The following figure shows an example of how values can be scaled:



Syntax

The following syntax is used for the "Scale" instruction:

SCL

```
SCALE_X(MIN := <Operand>,
VALUE := <Operand>,
 MAX := <Operand>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
MIN	Input	Integers, floating-point numbers	Low limit of the value range
VALUE	Input	Floating-point numbers	Value to be scaled.
MAX	Input	Integers, floating-point numbers	High limit of the value range
Function value		Integers, floating-point numbers	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := SCALE_X(MIN := "Tag_Value1",
 VALUE := "Tag_Real",
 MAX := "Tag_Value2");
```

The result of the instruction is returned as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Real	0.5
Tag_Value1	10
Tag_Value2	30
Tag_Result	20

See also

Overview of the valid data types (Page 741)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

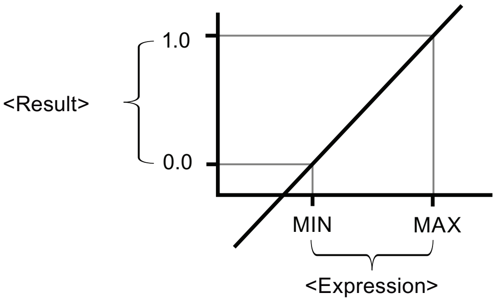
Editing SCL instructions (Page 1005)

NORM_X: Normalize

Description

The "Normalize" instruction normalizes a value by mapping it to a linear scale. You can use the parameters MIN and MAX to define the limits of a value range that is applied to the scale. Depending on the location of the value to be normalized in this value range, the result is calculated and saved as a floating-point number. If the value to be normalized is equal to the value at the MIN input, the instruction returns the value "0.0" as the result. If the value to be normalized is equal to the value at the MAX input, the instruction returns the result "1.0".

The following figure shows an example of how values can be normalized:



Syntax

The following syntax is used for the "Normalize" instruction:

SCL

```
NORM_X(MIN := <Operand>,
VALUE := <Operand>,
 MAX := <Operand>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
MIN	Input	Integers, floating-point numbers	Low limit of the value range
VALUE	Input	Integers, floating-point numbers	Value to be normalized.
MAX	Input	Integers, floating-point numbers	High limit of the value range
Function value		Floating-point numbers	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := NORM_X(MIN := "Tag_Value1",
 VALUE := "Tag_InputValue",
 MAX := "Tag_Value2");
```

The result of the instruction is returned as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InputValue	20
Tag_Value1	10
Tag_Value2	30
Tag_Result	0.5

See also

Overview of the valid data types (Page 741)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

Program control operations

IF: Run conditionally

Description

The "Run conditionally" instruction branches the program flow based on a condition. The condition is an expression with Boolean value ((TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

Syntax

Depending on the type of branch, you can program the following forms of the instruction:

- Branch through IF:

SCL

```
IF <Condition> THEN <Instructions>
END_IF
```

If the condition is satisfied, the instructions programmed after the THEN are executed. If the condition is not satisfied, the execution of the program continues with the next instruction after the END_IF.

- Branch through IF and ELSE:

SCL

```
IF <Condition> THEN <Instructions1>
ELSE <Instructions0>;
END_IF
```

If the condition is satisfied, the instructions programmed after the THEN are executed. If the condition is not satisfied, the instructions programmed after the ELSE are executed. Then the execution of the program continues with the next instruction after the END_IF.

- Branch through IF, ELSIF and ELSE:

SCL

```
IF <Condition1> THEN <Instructions1>
ELSIF <Condition2> THEN <Instruction2>
ELSE <Instructions0>;
END_IF;
```

If the first condition (<Condition1>) is satisfied, the instructions (<Instructions1>) after the THEN are executed. After execution of the instructions, the execution of the program continues after the END_IF.

If the first condition is not satisfied, the second condition (<Condition2>) is checked. If the second condition (<Condition2>) is fulfilled, the instructions (<Instructions2>) after the THEN are executed. After execution of the instructions, the execution of the program continues after the END_IF.

If none of the conditions are fulfilled, the instructions (<Instructions0>) after ELSE are executed followed by the execution of the program after END_IF.

You can nest as many combinations of ELSIF and THEN as you like within the IF instruction. The programming of an ELSE branch is optional.

The syntax of the IF instruction consists of the following parts:

Part	Data type	Description
<Condition>	BOOL	Expression to be evaluated
<Instructions>	-	Instructions to be executed with satisfied condition. An exception are instructions programmed after the ELSE. These are executed if no condition within the program loop is satisfied.
Function value	INT	Current value of the expression

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := IF "Tag_1" = 1 THEN "Tag_Value" := 10;
 ELSIF "Tag_2" = 1 THEN "Tag_Value" := 20;
 ELSIF "Tag_3" = 1 THEN "Tag_Value" := 30;
 ELSE "Tag_Value" := 0;
 END_IF;
```

The current value of the expression is stored in the "Tag_Value" operand and returned in the INT format as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_1	1	0	0	0
Tag_2	0	1	0	0
Tag_3	0	0	1	0
Tag_Value	10	20	30	0

See also

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

Overview of the valid data types (Page 741)

CASE: Run distributed

Description

The "Run distributed" instruction executes one of several instruction sequences depending on the value of a numerical expression.

The value of the expression must be an integer. When the instruction is executed, the value of the expression is compared with the values of several constants. If the value of the expression agrees with the value of a constant, the instructions programmed directly after this constant are executed. The constants can assume the following values:

- An integer (for example, 5)
- A range of integers (for example, 15..20)
- An enumeration consisting of integers and ranges (for example, 10, 11, 15..20)

Syntax

The following syntax is used for the "Run distributed" instruction:

SCL

```
CASE <expression> OF
<Constant1>: <Instructions1>
<Constant2>: <Instructions2>
<ConstantX>: <InstructionsX>; // X >=3
ELSE <Instructions0>;
END_CASE
```

The syntax of the instruction consists of the following parts:

Part/Parameter	Data type	Description
<expression>	Integers	Value which is compared to the programmed constant values.
<Constant>	Integers	Constant values which form the condition for the execution of an instruction sequence. The constants can assume the following values: • An integer (for example, 5) • A range of integers (for example, 15..20) • An enumeration consisting of integers and ranges (for example, 10, 11, 15..20)
<Instruction>	-	Any instructions which are executed if the value of the expression agrees with the value of a constant. An exception are instructions programmed after the ELSE. These instructions are executed if the values do not agree.
Function value	INT	Current value of the expression

For additional information on valid data types, refer to "See also".

If the value of the expression agrees with the value of the first constant (<Constant1>), the instructions (<Instructions1>) which are programmed directly after the first constant are executed. Program execution subsequently resumes after the END_CASE.

If the value of the expression does not agree with the value of the first constant (<Constant1>), this value is compared to the value of the constant which is programmed next. In this way, the CASE instruction is executed until the values agree. If the value of the expression does not correspond to any of the programmed constant values, the instructions (<Instructions0>) which are programmed after the ELSE are executed. ELSE is an optional part of the syntax and can be omitted.

The CASE instruction can also be nested by replacing an instruction block with CASE . END_CASE represents the end of the CASE instruction.

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := CASE "Tag_Value" OF
 0 : "Tag_1" := 1;
 1,3,5 : "Tag_2" :=1;
 6..10 : "Tag_3" := 1;
 16,17,20..25 : "Tag_4" := 1;
 ELSE "Tag_5" := 1;
 END_CASE;
```

The current value of the expression is stored in the "Tag_Value" operand and returned in the INT format as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Values				
Tag_Value	0	1, 3 , 5	6, 7, 8, 9, 10	16,17, 20, 21, 22, 23, 24, 25	2
Tag_1	1	-	-	-	-
Tag_2	-	1	-	-	-
Tag_3	-	-	1	-	-
Tag_4	-	-	-	1	-
Tag_5	-	-	-	-	1
1: The operand is set to the signal state "1". -: The signal state of the operand remains unaltered.					

See also

CONTINUE: Recheck loop condition (Page 1641)

Overview of the valid data types (Page 741)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

EXIT: Exit loop immediately (Page 1642)

FOR: Run in counting loop

Description

The "Run in counting loop" instruction causes repeated execution of a program loop until a run variable lies within a specified value range.

Program loops can also be nested. Within a program loop, you can program additional program loops with other run variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). Use the instruction "Exit loop immediately" (EXIT) to exit the entire loop execution. For additional information on this topic refer to "See also."

Syntax

The following syntax is used for the "Run in counting loop" instruction:

SCL

```
FOR <Run_tag> := <Start_value> TO <End_value> BY <Increment> DO <Instructions>
END_FOR
```

The syntax of the FOR instruction consists of the following parts:

Part	Data type	Description
<Run tag>	Integers	Operand whose value is evaluated with the loop execution.
<Start value>	Integers	Expression whose value is allocated at the start of the loop execution of the run tags.
<End value>	Integers	Expression whose value defines the last run of the program loop. After each loop run there is a check to determine if the run tag has reached the end value. If the run variable has not reached the end value, the instructions that are programmed after the DO are executed. If the end value has been reached, program continues after END_FOR. An alteration to the end value is not permitted during execution of the instruction.

Part	Data type	Description
<Increment>	Integers	Expression by whose value the run variable is increased (positive increment) or decreased (negative increment) after each loop. Specification of the increment is optional. If no increment is given, the value of the run tag is increased by 1 after each loop. An alteration of the increment is not permitted during execution of the instruction.
<Instructions>	-	Instructions which are carried out with each loop, as long as the value of the run tag lies within the value range. The value range is defined by the start and end values.
Function value	INT	Current value of the expression

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := FOR i := 2 TO 8 BY 2
 DO "a_array[i] := "Tag_Value"*"b_array[i]";
 END_FOR;
```

The "Tag_Value" operand is multiplied with the components (2, 4, 6, 8) of the "b_array" ARRAY tag. The result is read into the components (2, 4, 6, 8) of the "a_array" ARRAY tag. The current value of the expression is stored in the "Tag_Value" operand and returned in the INT format as a function value.

See also

CONTINUE: Recheck loop condition (Page 1641)

EXIT: Exit loop immediately (Page 1642)

Overview of the valid data types (Page 741)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

WHILE: Run if condition is met

Description

The "Run if condition is met" instruction causes a program loop to be repeatedly executed until the implementation condition is fulfilled. The condition is an expression with Boolean value ((TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

Program loops can also be nested. Within a program loop, you can program additional program loops with other run variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). Use the instruction "Exit loop immediately" (EXIT) to exit the entire loop execution. For additional information on this topic refer to "See also."

Syntax

The following syntax is used for the "Run if condition is met" instruction:

```
SCL
WHILE <Condition> DO <Instructions>
END_WHILE
```

The syntax of the WHILE instruction consists of the following parts:

Part	Data type	Description
<Condition>	BOOL	Expression which is evaluated before each loop.
<Instructions>	-	Instructions to be executed with satisfied condition. If the condition has not been satisfied, program execution continues after END_WHILE.
Function value	INT	Current value of the expression

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := WHILE "Tag_Value1" <> "Tag_Value2"
 DO "Tag_Result" := "Tag_Input";
 END_WHILE;
```

As long as the values of the "Tag_Value1" and "Tag_Value2" operands do not match, the value of the "Tag_Result" operand is allocated to the "Tag_Input" operand. The current value of the expression is stored in the "Tag_Value" operand and returned in the INT format as a function value.

See also

EXIT: Exit loop immediately (Page 1642)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

CONTINUE: Recheck loop condition (Page 1641)

Overview of the valid data types (Page 741)

REPEAT: Run if condition is not met

Description

The "Run if condition is not met" instruction causes a program loop to be repeatedly executed as long as the termination condition is not satisfied. The condition is an expression with Boolean value ((TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

The instructions are executed once, even if the termination condition is fulfilled.

Program loops can also be nested. Within a program loop, you can program additional program loops with other run variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). Use the instruction "Exit loop immediately" (EXIT) to exit the entire loop execution. For additional information on this topic refer to "See also."

Syntax

The following syntax is used for the "Run if condition is not met" instruction:

```
SCL
REPEAT <Instructions>
UNTIL <Condition> END_REPEAT
```

The syntax of the REPEAT instruction consists of the following parts:

Part	Data type	Description
<Instructions>	-	Instructions that are executed as long as the programmed condition has the value FALSE. The instructions are executed once, even if the termination condition is fulfilled.
<Condition>	BOOL	Expression which is evaluated after each loop. If the expression has the value FALSE, the program loop is executed once again. If the expression has the value TRUE, the program loop continues after END_REPEAT.
Function value	INT	Current value of the expression

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := REPEAT "Tag_Result" := "Tag_Value";
 UNTIL "Tag_Error" END_REPEAT;
```

As long as the value of the "Tag_Error" operand has the signal state "0", the value of the "Tag_Value" operand is allocated to the "Tag_Result" operand. The current value of the expression is stored in the "Tag_Value" operand and returned in the INT format as a function value.

See also

CONTINUE: Recheck loop condition (Page 1641)

EXIT: Exit loop immediately (Page 1642)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

Overview of the valid data types (Page 741)

CONTINUE: Recheck loop condition

Description

The "Recheck loop condition" instruction ends the current program run of a FOR, WHILE or REPEAT loop.

After execution of the instruction, the conditions for the continuation of the program loop are evaluated again. The instruction affects the program loop which directly contains the instruction.

Syntax

The following syntax is used for the "Recheck loop condition" instruction:

SCL

```
CONTINUE
```

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := FOR i := 1 TO 15 BY 2 DO
 IF (i < 5) THEN
 CONTINUE;
 END_IF;
 DB10.DB[i] := 1;
 END_FOR;
```

If the i < 5 condition is fulfilled, the subsequent value allocation (DB10.DB[i] := 1) is not executed. The run variable (i) is increased by the increment of "2" and there is a check as to whether its current value lies within the programmed value range. If the run variable lies in the value range, the IF condition is evaluated again.

If the i < 5 condition is not fulfilled, the subsequent value allocation (DB10.DB[i] := 1) is executed and a new loop started. In this case, the run variable is also increased by the increment "2" and checked.

The current value of the expression is returned in the INT format as a function value.

See also

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

EXIT: Exit loop immediately (Page 1642)

EXIT: Exit loop immediately

Description

The instruction "Exit loop immediately" cancels the execution of a FOR, WHILE or REPEAT loop at a certain point regardless of conditions. The execution of the program is continued after the end of the loop (END_FOR, END_WHILE, END_REPEAT).

The instruction affects the program loop which directly contains the instruction.

Syntax

The following syntax is used for the "Exit loop immediately" instruction:

SCL

```
EXIT
```

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := FOR i := 15 TO 1 BY -2 DO
 IF (i < 5)
 THEN EXIT;
 END_IF;
 DB10.DB[i] := 1;
 END_FOR;
```

If the i < 5 condition is fulfilled, the execution of the loop is cancelled. Program execution resumes after the END_FOR.

If the i < 5 condition is not fulfilled, the subsequent value allocation (DB10.DB[i] :=1) is executed and a new loop started. The run tag (i) is decreased by the increment of "-2" and it is checked whether its current value lies in the programmed value range. If the (i) run variable lies within the value range, the IF condition is evaluated again.

The current value of the expression is returned in the INT format as a function value.

See also

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

CONTINUE: Recheck loop condition (Page 1641)

GOTO: Jump

Description

The "Jump" instruction resumes the execution of a program at a given point marked as a jump label.

The jump labels and the "Jump" instruction must be in the same block. The name of a jump label can only be assigned once within a block. Each jump label can be the target of several jump instructions.

A jump from the "outside" into a program loop is not permitted, but a jump from a loop to the "outside" is possible.

Syntax

Use the following syntax for the "Jump" instruction:

```
SCL
GOTO <Jump label>
...
<Jump label>: <Instructions>
```

The syntax of the GOTO instruction consists of the following parts:

Part / Parameter	Description
<jump label>	Jump label to be jumped to
<Instructions>	Instructions which are executed after the jump.
Function value	Current value of the expression

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := CASE "Tag_Value" OF
```

SCL

```
1 : GOTO MyLABEL1;
2 : GOTO MyLABEL2;
3 : GOTO MyLABEL3;
ELSE GOTO MyLABEL4;
END_CASE;
MyLABEL1: "Tag_1" := 1;
MyLABEL2: "Tag_2" := 1;
MyLABEL3: "Tag_3" := 1;
MyLABEL4: "Tag_4" := 1;
```

Depending on the value of the "Tag_Value" operand, the execution of the program resumes at the point identified by the corresponding jump label. If the "Tag_Value" operand has the value 2, for example, program execution resumes at the jump label "MyLABEL2". The program line identified by the "MyLABEL1" jump label is skipped in this case.

The current value of the expression is stored in the "Tag_Value" operand and returned in the INT format as a function value.

See also

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

RETURN: Exit block

Description

Use the "Exit block" instruction to end the program execution in the block currently being processed and resume in the calling block.

The instruction can be omitted at the end of the block.

Syntax

Use the following syntax for the "Exit block" instruction:

SCL

```
RETURN
```

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := IF "Tag_Error" <>0 THEN RETURN;
 END_IF;
```

If the signal state of the "Tag_Error" operand is zero, execution of the program ends in the block currently being processed. The current value of the expression is returned in the INT format as a function value.

See also

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

Runtime control

STP: Exit program

Description

The "Exit program" instruction sets the CPU to STOP mode and terminates the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

Syntax

The following syntax is used for the "Exit program" instruction:

SCL

```
STP()
```

See also

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

RE_TRIGR: Restart cycle monitoring time

Description

The "Restart cycle monitoring time" instruction restarts the cycle monitoring of the CPU. The monitoring of the cycle time then restarts with the period you have set in the CPU configuration. By restarting the cycle monitoring time, you prevent errors from being triggered or setting the CPU to STOP.

The "Restart cycle monitoring time" instruction can be used in the block of priority class 1 (for example, the cyclic OB) and in the blocks called within it.

If the instruction is called in a block with a higher priority, for example, a hardware interrupt, diagnostic interrupt or cyclic interrupt, the instruction is not be executed.

Syntax

The following syntax is used for the "Restart cycle monitoring time" instruction:

SCL
`RE_TRIGR()`

See also

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

GetError: Get error locally

Description

You can use the "Get error locally" instruction to query the occurrence of errors within a block. If the system signals errors during block execution, the instruction gives detailed information about the first error that occurred.

The error information can only be saved in operands of the "ErrorStruct" system data type. The system data type "ErrorStruct" specifies the exact structure in which the information about the error is stored. Use additional instructions to evaluate this structure and program an appropriate response. When the first error has been eliminated, the instruction issues additional information about the next error that has occurred.

Syntax

Use the following syntax for the "Get error locally" instruction:

```
SCL
<Error information> := GET_ERROR()
```

The syntax of the instruction consists of the following parts:

Part/Parameter	Declaration	Data type	Description
<Error information>	Output	ErrorStruct	Information about errors that have occurred

Data type "ErrorStruct"

The following table shows the structure of the data type ErrorStruct:

<table>
<tr><th colspan="2">Structure components</th><th>Data type</th><th colspan="6">Description</th></tr>
<tr><td colspan="2">ERROR_ID</td><td>WORD</td><td colspan="6">Error ID</td></tr>
<tr><td colspan="2">FLAGS</td><td>BYTE</td><td colspan="6">Shows if an error occurred during a block call.
16#01: Error during a block call.
16#00: No error during a block call.</td></tr>
<tr><td colspan="2">REACTION</td><td>BYTE</td><td colspan="6">Default reaction:
0: Ignore (write error),
1: Continue with substitute value "0" (read error),
2: Skip instruction (system error)</td></tr>
<tr><td colspan="2">CODE_ADDRESS</td><td>CREF</td><td colspan="6">Information on address and type of block</td></tr>
<tr><td></td><td>BLOCK_TYPE</td><td>BYTE</td><td colspan="6">Type of block where the error occurred:
1: OB
2: FC
3: FB</td></tr>
<tr><td></td><td>CB_NUMBER</td><td>UINT</td><td colspan="6">Number of the code block</td></tr>
<tr><td></td><td>OFFSET</td><td>UDINT</td><td colspan="6">Reference to the internal memory</td></tr>
<tr><td colspan="2" rowspan="7">MODE</td><td rowspan="7">BYTE</td><td colspan="6">Access mode: Depending on the type of access, the following information can be output:</td></tr>
<tr><td>Mode</td><td>(A)</td><td>(B)</td><td>(C)</td><td>(D)</td><td>(E)</td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td><td>Offset</td></tr>
<tr><td>2</td><td></td><td></td><td>Area</td><td></td><td></td></tr>
<tr><td>3</td><td>Location</td><td>Scope</td><td></td><td>Numb er</td><td></td></tr>
<tr><td>4</td><td></td><td></td><td>Area</td><td></td><td>Offset</td></tr>
</table>

Structure components		Data type	Description					
			5			Area	DB no.	Offset
			6	PtrNo./Acc		Area	DB no.	Offset
			7	PtrNo./Acc	Slot No. / Scope	Area	DB no.	Offset
OPERAND_NUMBER		UINT	Operand number of the machine command					
POINTER_NUMBER_ LOCATION		UINT	(A) Internal pointer					
SLOT_NUMBER_SCOPE		UINT	(B) Storage area in internal memory					
DATA_ADDRESS		NREF	Information about the address of an operand					
	AREA	BYTE	(C) Memory area: L: 16#40 – 4E, 86, 87, 8E, 8F, C0 – CE E: 16#81 A: 16#82 M: 16#83 DB: 16#84, 85, 8A, 8B					
	DB_NUMBER	UINT	(D) Number of the data block					
	OFFSET	UDINT	(E) Relative address of the operand					

Structure component "ERROR_ID"

The following table shows the values that can be output at the structure component "ERROR_ID":

ID (hexadecimal)	ID (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2505	9477	Calling the instruction "Stop" (SFC46) in the user program
2520	9504	Invalid STRING
2522	9506	Read errors: Operand outside the valid range
2523	9507	Write errors: Operand outside the valid range
2524	9508	Read errors: Invalid operand
2525	9509	Write errors: Invalid operand
2528	9512	Read errors: Data alignment
2529	9513	Write errors: Data alignment
252C	9516	Invalid pointer
2530	9520	Write errors: Data block
2533	9523	Invalid pointer used

ID (hexadecimal)	ID (decimal)	Description
2534	9524	Block number error FC
2535	9525	Block number error FB
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2942	10562	Read errors: Input
2943	10563	Write errors: Output

The instruction "Get error locally" can also be used to forward an alarm about the error status to the calling block. To do this, you have to program the instruction at the end of the called block.

Note

The instruction "Get error locally" enables local error handling within a block. When "Get error locally" is inserted in the program code of a block, any predefined system responses are ignored if an error occurs.

See also

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

GetErrorID: Get error ID locally

Description

You can use the "Get error ID locally" instruction to query the occurrence of errors within a block. If the system signals errors during block execution, the error ID of the first error that has occurred is given. The error ID can only be saved in operands of the WORD data type. When the first error has been eliminated, the instruction outputs the error ID of the next error that occurred.

The instruction "Get error ID locally" can also be used to forward an alarm about the error status to the calling block. To do this, you have to program the instruction at the end of the called block.

Note

The instruction "Get error ID locally" enables local error handling within a block. When "Get error ID locally" is inserted in the program code of a block, any predefined system responses are ignored if an error occurs.

Syntax

Use the following syntax for the "Get error ID locally" instruction:

```
SCL
<Error_ID> := GET_ERR_ID()
```

The syntax of the instruction consists of the following parts:

Part/Parameter	Declaration	Data type	Description
<Error_ID>	Output	WORD	Error ID

Error ID

The following table shows the values that can be output:

ID (hexadecimal)	ID (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2505	9477	Calling the instruction "Stop" (SFC46) in the user program
2520	9504	Invalid STRING
2522	9506	Read errors: Operand outside the valid range
2523	9507	Write errors: Operand outside the valid range

ID (hexadecimal)	ID (decimal)	Description
2524	9508	Read errors: Invalid operand
2525	9509	Write errors: Invalid operand
2528	9512	Read errors: Data alignment
2529	9513	Write errors: Data alignment
252C	9516	Invalid pointer
2530	9520	Write errors: Data block
2533	9523	Invalid pointer used
2534	9524	Block number error FC
2535	9525	Block number error FB
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2942	10562	Read errors: Input
2943	10563	Write errors: Output

See also

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

Word logic operations

DECO: Decode

Description

The "Decode" instruction sets a bit specified by the input value in the output value.

The "Decode" instruction reads the value of the IN parameter and sets the bit in the result value, whose bit position corresponds to the value read. The other bits in the output value are filled with zeroes. If the value of the IN parameter is greater than 31, a modulo 32 instruction is executed.

Syntax

The following syntax is used for the "Decode" instruction:

```
SCL
DECO(IN := <Expression>),
OUT => <Result>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
<expression>	Input	Integers	Position of the bit in the output value which is set.
Function value		Bit strings	Current output value

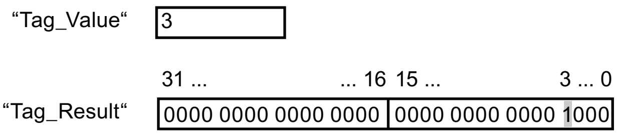
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
DECO(IN := "Tag_Value"),
     OUT => "Tag_Result");
```

The following figure shows how the instruction works using specific operand values:



The instruction reads bit number "3" from the value of the "Tag_Value" operand and sets the third bit to the value of the "Tag_Result" operand. The current output value is returned in the "Tag_Value" operand and as a function value.

See also

Overview of the valid data types (Page 741)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

ENCO: Encode

Description

The "Encode" instruction reads the bit number of the least significant bit set in the input value and outputs it as result.

Syntax

The following syntax is used for the "Encode" instruction:

```
SCL
ENCO(IN := <Expression>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
<Expression>	Input	Bit strings	Input value
Function value		INT	Result of the instruction

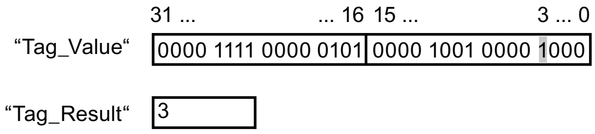
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := ENCO(IN := "Tag_Value");
```

The following figure shows how the instruction works using specific operand values:



The instruction selects the least significant set bit of the "Tag_Value" operand and writes the bit position "3" to the "Tag_Result" operand. The result of the instruction is returned in the "Tag_Value" operand and as a function value.

See also

Overview of the valid data types (Page 741)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

SEL: Select

Description

Depending on a switch (G input), the "Select" instruction selects one of the inputs, IN0 or IN1 and outputs its content as the result. If the G input has the signal state "0", the value at the IN0 input is moved. If the G input has the signal state "1", the value at the IN1 input is moved and returned as a function value.

The instruction can only be executed if the tags of all parameters of the data type have the same class.

Syntax

The following syntax is used for the "Select" instruction:

SCL

```
SEL(G := <Expression>,
IN0 := <expression>,
IN1 := <Expression>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
G	Input	BOOL	Switch
IN0	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, characters	First input value
IN1	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, characters	Second input value
Function value		Bit strings, integers, floating-point numbers, TIME, TOD, DATE, characters	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
 SCL
"Tag_Result" := SEL(G := "Tag_Value",
                    IN0 := "Tag_0",
                    IN1 := "Tag_1");
```

The result of the instruction is stored in the "Tag_Value" operand and returned as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	0	1
Tag_0	W#16#0000	W#16#4C
Tag_1	W#16#FFFF	D #16#5E
Tag_Result	W#16#0000	D #16#5E

See also

Overview of the valid data types (Page 741)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

MUX: Multiplex

Description

The "Multiplex" instruction moves the value of a selected input and sends it. The K parameter determines the number of the input whose value should be moved. Numbering starts at IN0 and is incremented continuously with each new input. You can declare a maximum of 32 input parameters.

If the value of the parameter K is greater than the number of inputs, and the parameter INELSE is not set, the function value of the instruction is invalid and the enable output ENO is set to "0".

Numeric data types and time data types are permitted at the inputs. All tags with parameters assigned must be of the same data type. The function assumes the most significant data type.

Syntax

The following syntax is used for the "Multiplex" instruction:

```
 SCL
MUX(K := <Expression>,
    IN0 := <Expression>,
    IN1 := <Expression>,
    INELSE := <Expression>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
K	Input	Integers	Specifies the parameter whose content is to be transferred.
IN0	Input	Binary numbers, bit strings, integers, floating-point numbers, times, characters, TOD, DATE	First input value
IN1	Input	Binary numbers, bit strings, integers, floating-point numbers, times, characters, TOD, DATE	Second input value
INn	Input	Binary numbers, bit strings, integers, floating-point numbers, times, characters, TOD, DATE	Optional input values
INELSE	Input	Binary numbers, bit strings, integers, floating-point numbers, times, characters, TOD, DATE	Specifies the value to be copied when K > n.
Function value		Binary numbers, bit strings, integers, floating-point numbers, times, characters, TOD, DATE	Operand to which the selected input value is transferred.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
 SCL
"Tag_Result" := MUX(K := "Tag_Value",
                    IN0 := "Tag_0",
                    IN1 := "Tag_1",
                    INELSE := "Tag_2");
```

The result of the instruction is returned in the "Tag_Value" operand and as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	1	4
Tag_0	DW#16#00000000	DW#16#00000000
Tag_1	DW#16#3E4A7D	DW#16#3E4A7D
Tag_2	DW#16#FFFF0000	DW#16#FFFF0000
Tag_Result	DW#16#3E4A7D	DW#16#FFFF0000

See also

Overview of the valid data types (Page 741)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

DEMUX: Demultiplex

Description

The "Demultiplex" instruction transfers the value of the IN input parameter to a selected output parameter. The selection of the input parameter takes place independently of the parameter value K. The K parameter specifies the output parameter number to which the value of the IN parameter is transferred. The other outputs are not changed. Numbering starts with IN0 and is incremented continuously with each new output. You can declare a maximum of 32 output parameters.

If the value of the K parameter is greater than the number of output parameters, the value of the IN parameter is transferred to the ELSE parameter.

Syntax

The following syntax is used for the "Demultiplex" instruction:

```
SCL
DEMUX (K := <expression>,
 IN := <expression>,
 OUT0 := <operand>,
 OUT1 := <operand>,
 ELSE := <operand>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
K	Input	UINT	Specifies the number of the output parameter into which the input value (IN)) is transferred.
IN	Input	Bit strings, integers, floating-point numbers, CHAR, TIME	Input value
OUT0	Output	Bit strings, integers, floating-point numbers, CHAR, TIME	First output
OUT1	Output	Bit strings, integers, floating-point numbers, CHAR, TIME	Second output
OUTn	Output	Bit strings, integers, floating-point numbers, CHAR, TIME	Optional outputs
ELSE	Output	Bit strings, integers, floating-point numbers, CHAR, TIME	Output to which the input value (IN at K > n is transferred.
Function value		Bit strings, integers, floating-point numbers, CHAR, TIME	Result of the instruction

For additional information on available data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := DEMUX(K := "Tag_Value",
 IN := "Tag_Value_1",
 OUT0 := "Tag_0",
 OUT1 := "Tag_1",
 ELSE := "Tag_2");
```

The result of the instruction is returned in the "Tag_Value" operand and as a function value.

The following tables show how the instruction works using specific operand values:

Table 9- 43 Input values of the "Demultiplex" instruction before the network execution

Parameters	Operand	Values	
K	Tag_Number	1	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#3E4A7D

Table 9- 44 Output values of the "Demultiplex" instruction after the network execution

Parameters	Operand	Values	
OUT0	Tag_Output_0	Unchanged	Unchanged
OUT1	Tag_Output_1	DW#16#FFFFFFFF	Unchanged
ELSE	Tag_Output_2	Unchanged	DW#16#3E4A7D

See also

Overview of the valid data types (Page 741)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

Shift and rotate

SHR: Shift right

Description

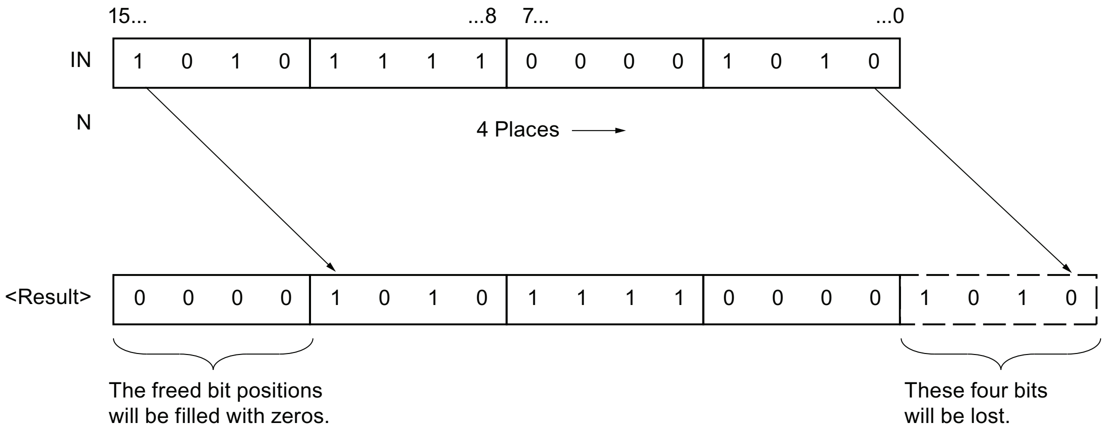
The "Shift right" instruction shifts the content of the IN parameter bit-by-bit to the right and returns it as function value. The parameter N is used to specify the number of bit positions by which the specified value should be shifted.

If the value of the N parameter is "0", the value of the IN parameter is given as a result.

If the value of the N parameter is greater than the number of bit positions, the value of the IN parameter is shifted to the right by the available number of bit positions.

The bit positions that are freed by shifting in the left operand area are filled with zeros.

The following figure shows how the content of an integer data type operand is shifted by four bit positions to the right:



Syntax

The following syntax is used for the "Shift right" instruction:

```
SCL
SHR(IN := <Operand>,
N := <Operand>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
IN	Input	Bit strings	Value that is shifted.
N	Input	Integers	Number of bits by which the value (IN) is shifted.
Function value		Bit strings	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := SHR (IN := "Tag_Value",
 N := "Tag_Number");
```

The result of the instruction is returned in the "Tag_Value" operand and as a function value.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	0011 1111 1010 1111
N	Tag_Number	3
Function value	Tag_Result	0000 0111 1111 010 1

The content of the "Tag_Value" operand is shifted by three bit positions to the right. The result is saved in the "Tag_Result" operand.

See also

Overview of the valid data types (Page 741)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

SHL: Shift left

Description

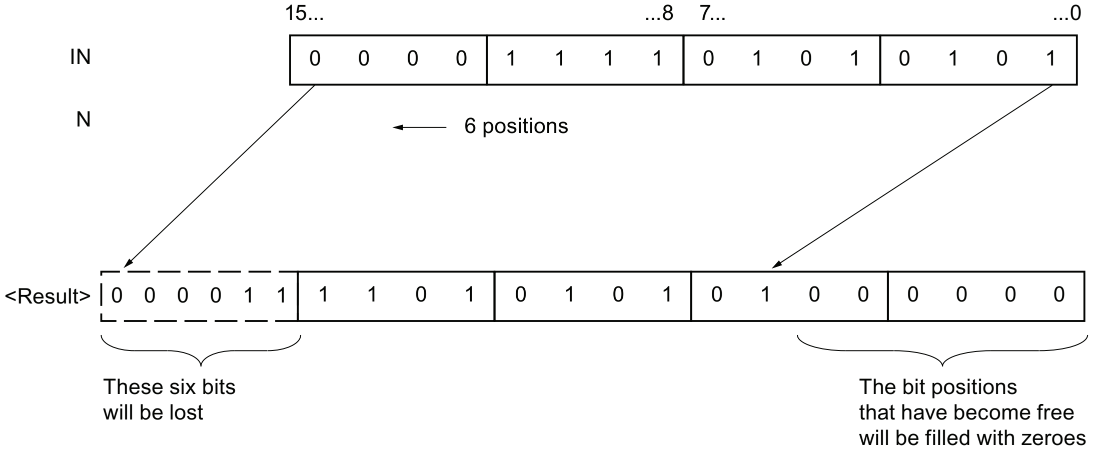
The "Shift left" instruction shifts the content of the IN parameter bit-by-bit to the left and returns it as function value. The parameter N is used to specify the number of bit positions by which the specified value should be shifted.

If the value of the N parameter is "0", the value of the IN parameter is given as a result.

If the value of the N parameter is greater than the number of bit positions, the value of the IN parameter is shifted to the left by the available number of bit positions.

The freed positions are filled with zeros in the result value.

The following figure shows how the content of an operand of the WORD data type is shifted by six bit positions to the left:



Syntax

The following syntax is used for the "Shift left" instruction:

SCL

```
SHL(IN := <operand>,
 N := <Operand>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
IN	Input	Bit strings	Value that is shifted.
N	Input	Integers	Number of bits by which the value (IN) is shifted.
Function value		Bit strings	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := SHL(IN := "Tag_Value",
                    N := "Tag_Number");
```

The result of the instruction is returned in the "Tag_Value" operand and as a function value.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	0011 1111 1010 1111
N	Tag_Number	4
Function value	Tag_Result	1111 1010 1111 0000

The value of the "Tag_Value" operand is shifted by four bit positions to the left. The result is saved in the "Tag_Result" operand.

See also

Overview of the valid data types (Page 741)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

ROR: Rotate right

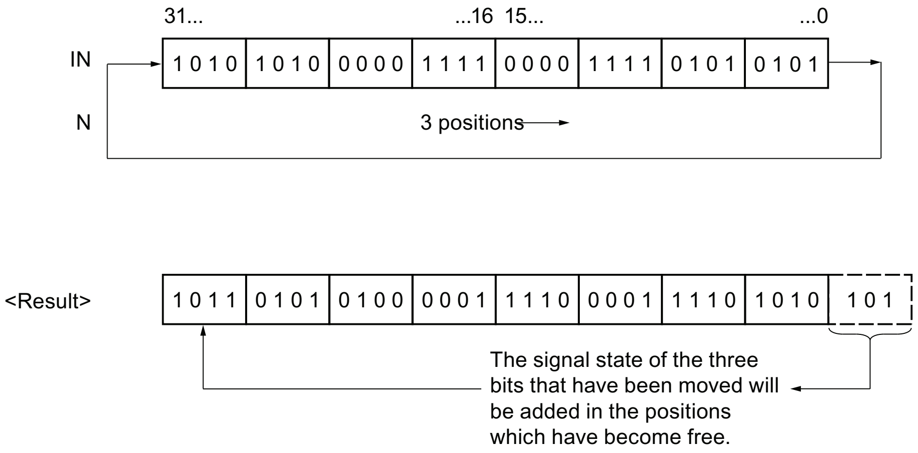
Description

The "Rotate right" instruction rotates the content of the IN parameter bit-by-bit to the right and assigns the result to the specified operand. The parameter N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value of the N parameter is "0", the value at input IN is given as a result.

If the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is still rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the right:



Syntax

The following syntax is used for the "Rotate right" instruction:

```
SCL
ROR(IN := <Operand>,
N := <Operand>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
IN	Input	Bit strings	Value that is rotated.
N	Input	Integers	Number of bit positions by which the (IN) value is rotated.
Function value		Bit strings	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := ROR(IN := "Tag_Value",
                    N := "Tag_Number");
```

The result of the instruction is returned in the "Tag_Value" operand and as a function value.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	0000 1111 1001 0101
N	Tag_Number	5
Function value	Tag_Result	1010 1000 0111 1100

The content of the "Tag_Value" operand is rotated by five bit positions to the right. The result is saved in the "Tag_Result" operand.

See also

Overview of the valid data types (Page 741)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

ROL: Rotate left

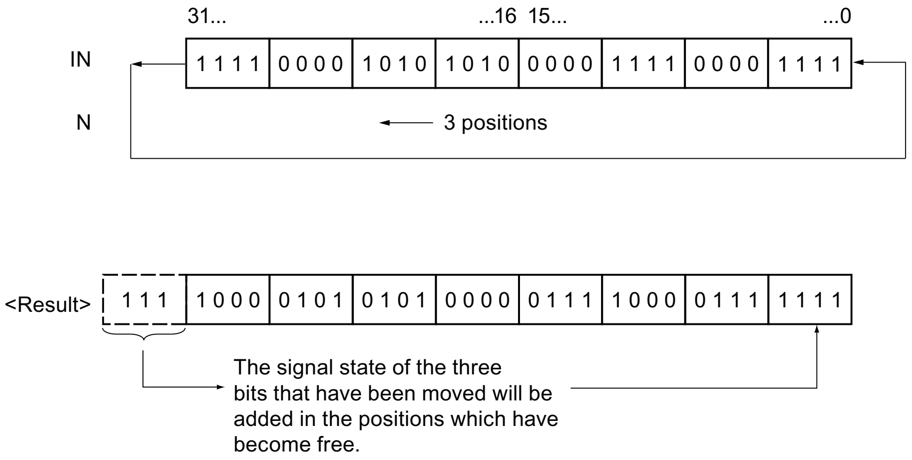
Description

The "Rotate left" instruction rotates the content of the IN parameter bit-by-bit to the left and returns it as function value. The parameter N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value of the N parameter is "0", the value at input IN is given as a result.

If the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is still rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the left:



The signal state of the three bits that have been moved will be added in the positions which have become free.

Syntax

The following syntax is used for the "Rotate left" instruction:

SCL

```
ROL(IN := <Operand>,
N := <Operand>)
```

The syntax of the instruction consists of the following parts:

Part / Parameter	Declaration	Data type	Description
IN	Input	Bit strings	Value that is rotated.
N	Input	Integers	Number of bit positions by which the value (IN) is rotated.
Function value		Bit strings	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
 SCL
"Tag_Result" := ROL(IN := "Tag_Value",
                    N := "Tag_Number");
```

The result of the instruction is returned in the "Tag_Value" operand and as a function value.

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1010 1000 1111 0110
N	Tag_Number	5
Function value	Tag_Result	0001 1110 1101 010 1

The content of the operand "Tag_Value" is rotated five bit positions to the left. The result is saved in the "Tag_Result" operand.

See also

Overview of the valid data types (Page 741)

Operators and operator precedence (Page 980)

Entering SCL instructions (Page 991)

Editing SCL instructions (Page 1005)

9.8.3 Extended instructions

9.8.3.1 Date and time-of-day

T_CONV: Convert times and extract

Description

You use this instruction to convert the data type of the IN input parameter to the data type that is output at the OUT output. You select the data formats for the conversion from the instruction boxes of the input and output.

Parameters

The following table shows the parameters of the "T_CONV" instruction. If an input and output parameter of the same data type is used, the instruction copies the corresponding value.

Parameter	Declaration	Data type	Memory area	Description
IN	Input	TIME, DINT	I, Q, M, D, L or constant	Value to be converted
OUT	Return	TIME, DINT	I, Q, M, D, L	Result of the conversion

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

T_ADD: Add times

Description

You use this instruction to add the time information in the IN1 input to the time information in the IN2 input. You can query the result in the OUToutput parameter. You can add the following formats:

- Addition of a time period (TIME) to another time period (TIME). The result can be output to a tag with the TIME format.
- Addition of a time period (TIME) to a time (DTL). The result can be output to a tag with the DTL format.

You decide the formats of the values in the IN1 input parameter and the OUT output parameter by selecting the data types for the input and output of the instruction. You can only specify time information with the TIME format in the IN2 input parameter.

Parameters

The following tables show the parameters of the "T_ADD" instruction, according to the possible conversions:

Table 9- 45 Addition of a time period (TIME) to another time period (TIME)

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	TIME	I, Q, M, D, L or constant	Summand
IN2	Input	TIME	I, Q, M, D, L or constant	Summand
OUT	Return	TIME	I, Q, M, D, L	Result of addition

Table 9- 46 Addition of a time period (TIME) to a time (DTL)

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	DTL	D	Summand
IN2	Input	TIME	I, Q, M, D, L or constant	Summand
OUT	Return	DTL	D	Result of addition

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

T_SUB: Subtract times

Description

You use this instruction to subtract the time information in the IN2 input parameter from the time information in the IN1 input parameter. You can query the difference in the OUT output parameter. You can subtract the following formats:

- Subtraction of a time period (TIME) from another time period (TIME). The result can be output to a tag with the TIME format.
- Subtraction of a time period (TIME) from a time (DTL). The result can be output to a tag with the DTL format.

You decide the formats of the values in the IN1 input parameter and the OUT output parameter by selecting the data types for the input and output parameters of the instruction. You can only specify time information with the TIME format in the IN2 input parameter.

Parameters

The following tables show the parameters of the "T_SUB" instruction, according to the possible conversions:

Table 9- 47 Subtraction of a time period (TIME) from another time period (TIME)

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	TIME	I, Q, M, D, L or constant	Minuend
IN2	Input	TIME	I, Q, M, D, L or constant	Subtrahend
OUT	Return	TIME	I, Q, M, D, L	Result of subtraction

Table 9- 48 Subtraction of a time period (TIME) from a time (DTL)

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	DTL	D	Minuend
IN2	Input	TIME	I, Q, M, D, L or constant	Subtrahend
OUT	Return	DTL	I, Q, M, D, L	Result of subtraction

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

T_DIFF: Time difference

Description

You use this instruction to subtract the time information in the IN2 input parameter from the time information in the IN1 input parameter. The result is stored in the OUT output parameter with the TIME format. Only values with the DTL format can be specified in the IN1 and IN2 input parameters.

If the time information in the IN2 input parameter is greater than the time information in the IN1 input parameter, the result is output as a negative value in the OUT output parameter.

Parameters

The following table shows the parameters of the "T_DIFF" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	DTL	D	Minuend
IN2	Input	DTL	D	Subtrahend
OUT	Return	TIME	I, Q, M, D, L	Difference in TIME format.

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

T_COMBINE: Combine times

Description

The instruction combines the DATE and TIME_OF_DAY (TOD) data formats and converts these formats into the DATE_AND_TIME (DTL) data format. The instruction does not report any errors.

- The DATE (IN1) input value must be between DATE#1990-01-01 and DATE#2089-12-31 (will not be checked).
- At the IN2 input value, the TIME_OF_DAY (TOD) data type is used.
- At the OUT output value, the DATE_AND_TIME (DTL) data type is output.

Parameter

The following table shows the parameters of the instruction "T_COMBINE":

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	DATE	D, L	Input tag in the DATE format.
IN2	Input	TOD	D, L	Input tag in the TIME_OF_DAY format.
OUT	Return	DTL	D	Return value in the DTL format.

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Time-of-day functions

WR_SYS_T: Set time-of-day

Description

You use this instruction to set the date and time-of-day of the CPU clock. You specify the date and time-of-day information with the DTL format in the IN input of the instruction. You can query whether errors have occurred during execution of the instruction in the RET_VAL output parameter.

The "WR_SYS_T" instruction cannot be used to pass information about the local time zone or daylight saving time.

Parameters

The following table shows the parameters of the "WR_SYS_T" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	DTL	D	Date and time-of-day
RET_VAL	Return	INT, REAL, DInt	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Error code (W#16#....)	Description
0000	No error
8081	Year invalid
8082	Month invalid
8083	Day invalid
8084	Hour information invalid
8085	Minute information invalid
8086	Second information invalid
8087	Nanosecond information invalid
80B0	The realtime clock has failed.

RD_SYS_T: Read time-of-day

Description

You use this instruction to read out the current date and current time-of-day of the CPU clock. The read-out dates are output in DTL format in the OUToutput parameter of the instruction. The provided value does not include information about the local time zone or daylight saving time. You can query whether errors have occurred during execution of the instruction in the RET_VALoutput.

Parameters

The following table shows the parameters of the "RD_SYS_T" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT, REAL, DInt	I, Q, M, D, L	Status of the instruction
OUT	Output	DTL	D	Date and time of CPU

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Error code (W#16#....)	Description
0000	No error
8222	The result is outside the permissible range of values
8223	The result cannot be saved with the specified data type

RD_LOC_T: Read local time

Description

You use this instruction to read out the current local time from the CPU clock and output this in DTLformat to the OUToutput. Information on the time zone and the start of daylight saving time and standard time, which you have set in the configuration of the CPU clock, are used to output the local time.

Parameters

The following table shows the parameters of the "RD_LOC_T" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	DINT, INT, LREAL, REAL	M, D, L	Status of the instruction
OUT	Output	DTL	D	Local time

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Error code (W#16#....)	Description
0000	No error
0001	No error. Local time is output as daylight saving time.
8080	Local time cannot be read out.

SET_TIMEZONE: Set time zone

Description

You use this instruction to calculate the local time based on the module time. The module time of the CPU is the UTC time. The module time is used exclusively for communication within the system. The rule for conversion to local time is defined in the "TimeTransformationRule" attribute that you specify in the TimeZone parameter. The rule defines the time zone calculation as well as the automatic changeover between daylight saving time and standard time.

Parameters

The following table shows the parameters of the "SET_TIMEZONE" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ=1: Conversion of module time to local time
TimeZone	Input	Time Transformation Rule (Page 1676)	D	Rule for conversion of module time to local time.
DONE	Output	BOOL	I, Q, M, D, L	• 0: Job not yet started or is still executing • 1: Job completed error-free
BUSY	Output	BOOL	I, Q, M, D, L	• 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	• 0: No error • 1: Error occurred
STATUS	Output	DINT, DWORD, UDINT, WORD	I, Q, M, D, L	Error message

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

STATUS parameter

Error code (W#16#....)	Description
7000	No job processing active.
7001	Start of the job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".

TimeTransformationRule

Description

The times for changeover to daylight saving time and standard time are defined in the TimeTransformationRule structure. The structure is as follows:

Name		Data type	Description
TimeTransformationRule		STRUCT	
	Bias	INT	// Time difference between local time and UTC [min]
	DaylightBias	INT	// Time difference between daylight saving and standard time [min]
	DaylightStartMonth	USINT	// Month of conversion to daylight saving time
	DaylightStartWeek	USINT	// Week of conversion to daylight saving time // 1 = First occurrence of the weekday in the month, ..., // 5 = Last occurrence of the weekday in the month
	DaylightStartWeekday	USINT	// Weekday of daylight saving time changeover: // 1 = Sunday
	DaylightStartHour	USINT	// Hour of daylight saving time changeover
	DaylightStartMinute	USINT	// Minute of daylight saving time changeover
	StandardStartMonth	USINT	// Month of conversion to standard time
	StandardStartWeek	USINT	// Week of conversion to standard time // 1 = First occurrence of the weekday in the month, ..., // 5 = Last occurrence of the weekday in the month
	StandardStartWeekday	USINT	// Weekday of standard time changeover: // 1 = Sunday
	StandardStartHour	USINT	// Hour of standard time changeover
	StandardStartMinute	USINT	// Minute of standard time changeover
	TimeZoneName	STRING[80]	// Name of time zone: "(GMT+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna"

RTM: Runtime meters

Description

You can use this instruction to set, start, stop, and read out a 32-bit runtime meter of your CPU.

You can write the values of the runtime meter to a memory card of the CPU by calling the instruction with the parameter MODE=7. This step uses the values that are current at the time the "RTM" instruction is executed. By saving the current values of the runtime meter, you can also transfer the current values of the runtime meter during transfer of the user program to a different CPU if they are required for the correct execution of your program.

Ensure that the runtime meter can also be stopped or restarted during execution of the user program, which may render the saved values incorrect.

Note

Avoid excessive read and write accesses to the memory card

Frequent read and write accesses to flash cards can reduce the service life of the card.

Parameters

The following table shows the parameters of the instruction "RTM":

Parameter	Declaration	Data type	Memory area	Description
NR	Input	RTM (UINT)	I, Q, M, D, L or constant	Number of the runtime meter Numbering starts with 0. For information on the number of runtime meters of your CPU, refer to the technical data.
MODE	Input	BYTE	I, Q, M, D, L or constant	Job ID: • 0: Read out (the status is then written to CQ and the current value to CV). After the runtime meter has reached (2E31) -1 hours, it stops at the highest value that can be displayed and outputs an "Overflow" error message. • 1: start (at the last counter value) • 2: stop • 4: set (to the value specified in PV) • 5: set (to the value specified in PV) and then start • 6: set (to the value specified in PV) and then stop • 7: Save all values of the runtime meter to the memory card (MC).

Parameter	Declaration	Data type	Memory area	Description
PV	Input	BYTE, DINT, INT, SINT, UINT, USINT, WORD	I, Q, M, D, L or constant	New value for the runtime meter
RET_VAL	Return	DINT, INT, LREAL, REAL	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
CQ	Output	BOOL	I, Q, M, D, L	Status of the runtime meter (1: running)
CV	Output	DINT	I, Q, M, D, L	Current value of the runtime meter

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8080	Wrong number for the runtime meter
8081	A negative value was passed to the PV parameter.
8082	Overflow of the runtime meter
8091	The MODE input parameter contains an illegal value.
80B1	The value cannot be written to the memory card (when called with parameter MODE=7).
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 1222)

9.8.3.2 String + Char

S_MOVE: Move character string

Description

You can use this instruction to move the content of a character string (STRING). The character string in the IN input parameter is copied to the OUT output parameter.

You can insert additional outputs for the S_MOVE instruction. In this case, the content of the operand in the IN input parameter is transferred to all available outputs.

You can use the "MOVE_BLK" and "UMOVE_BLK" instructions to copy tags of data type ARRAY.

Parameters

The following table shows the parameters of the "S_MOVE" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING	D, L	Source value
OUT	Output	STRING	D, L	Destination address

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

S_CONV: Convert character string

Description

You use this instruction to convert the value at the IN input to the data format you have specified in the OUT output. The following conversions are possible:

- Conversion of a character string (STRING) to a numerical value:

  The conversion is performed for all characters of the character string specified in the IN input parameter. Permitted characters are the digits "0" to "9", the decimal point, and the plus and minus signs. The first character of the string may be a valid number or a sign. Leading spaces and exponential notations are ignored.
  The character conversion can be interrupted by invalid characters. You decide the output format of the conversion by selecting a data type for the OUT output parameter.

- Conversion of a numerical value to a character string (STRING):

  You decide the format of the numeric value to be converted by selecting a data type for the IN input. A valid tag of the STRING data type must be specified in the OUT output. The length of the character string after conversion depends on the value at the IN input. The conversion result is saved as a string starting at the third byte. The first byte of the string records the maximum length and the second byte the actual length of the character string. Positive numeric value are output without a sign.

- Copying a character string:

  If you enter the STRING data type in the input and output parameters of the instruction, the character string in the IN input will be copied to the OUT output. If the actual length of the character string in the IN input exceeds the maximum character string length in the OUT output; only the part of the character string that exactly fits into the character string of OUT will be copied to IN.

  Note

  Exponential notation during conversion from floating-point numbers

  Do not use exponential notation ("e" or "E") for the conversion from floating-point numbers with the instruction "S_CONV". Instead, use the instruction "STRG_VAL (Page 1681)" for the conversion of floating-point numbers with exponential notation. You can use the FORMAT parameter of the instruction to select exponential notation as input format.

Parameters

The following tables show the parameters of the "S_CONV" instruction, according to the possible conversions:

Table 9- 49 Parameters for converting a character string to a numeric value:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	STRING	D, L	Value to be converted
OUT	Output	CHAR, USINT, UINT, UDINT, SINT, INT, DINT, REAL, LREAL	I, Q, M, D, L	Result of the conversion

Table 9- 50 Parameters for converting a numeric value to a character string:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	CHAR, USINT, UINT, UDINT, SINT, INT, DINT, REAL, LREAL	I, Q, M, D, L or constant	Value to be converted
OUT	Output	STRING	D, L	Result of the conversion

Table 9- 51 Parameters for copying a character string:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	STRING	D, L	Value to be copied
OUT	Output	STRING	D, L	Result of the copy operation

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

STRG_VAL: Convert character string to numerical value

Description

The "STRG_VAL" instruction converts a numeric character string to the corresponding integer or floating-point notation:

- You specify the character string to be converted in the IN input parameter.
- You define the format of the output value by selecting a data type for the OUT output parameter. You can query the result in the OUToutput parameter.

Permitted characters for the conversion are the digits "0" to "9", the decimal point, the decimal comma, notations "E" and "e", and the plus and minus characters. The conversion can be interrupted by invalid characters.

Parameters

The following table shows the parameters of the "STRG_VAL" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING	D, L	Numeric character string to be converted
FORMAT	Input	WORD	I, Q, M, D, L or constant	Output format of the characters
P	Input	UINT	I, Q, M, D, L	Reference to the first character to be converted (first character = 1, value "0" or a value > length of the string is invalid)
OUT	Output	USINT, SINT, UINT, INT, UDINT, DINT, REAL, LREAL	I, Q, M, D, L	Result of the conversion

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter FORMAT

You use the FORMAT parameter to specify how the characters of a character string are to be interpreted. Exponential values can also be converted and represented with the "STRG_VAL" instruction. Only tags of the USINT data type can be specified in the FORMAT parameter.

The following table shows the possible values of the FORMAT parameter and their meaning:

Value (W#16#....)	Notation	Decimal representation
0000	Decimal fraction	"."
0001		","
0002	Exponential	"."
0003		","
0004 to FFFF	Invalid values	

Parameter P

The conversion starts at the character whose position you specified in the P parameter. If, for example, the value "1" is specified in the P parameter, the conversion starts at the first character of the specified character string.

Example

The following table shows examples of the conversion of a character string to a numeric value:

IN (STRING)	FORMAT (W#16#....)	OUT (data type)	OUT (value)	ENO status
'123'	0000	INT/DINT	123	1
'-00456'	0000	INT/DINT	-456	1
'123.45'	0000	INT/DINT	123	1
'+2345'	0000	INT/DINT	2345	1
'00123AB'	0000	INT/DINT	123	1
'123'	0000	REAL	123.0	1
'-00456'	0001	REAL	-456.0	1
'+00456'	0001	REAL	456.0	1
'123.45'	0000	REAL	123.45	1
'123.45'	0001	REAL	12345.0	1
'123,45'	0000	REAL	12345.0	1
'123,45'	0001	REAL	123.45	1
'.00123AB'	0001	REAL	123.0	1
'1.23e-4'	0000	REAL	1.23	1
'1.23E-4'	0000	REAL	1.23	1
'1.23E-4'	0002	REAL	1.23E-4	1

IN (STRING)	FORMAT (W#16#....)	OUT (data type)	OUT (value)	ENO status
'12,345.67'	0000	REAL	12345.67	1
'12,345.67'	0001	REAL	12.345	1
'3.4e39'	0002	REAL	W#16#7F800000	1
'-3.4e39'	0002	REAL	W#16#FF800000	1
'1.1754943e-38'	0002	REAL	0.0	1
'12345'	-/-	SINT	0	0
'A123'	-/-	-/-	0	0
''	-/-	-/-	0	0
'++123'	-/-	-/-	0	0
'+-123'	-/-	-/-	0	0

VAL_STRG: Convert numerical value to character string

Description

You use this instruction to convert a numeric value to a character string.

- You specify the value to be converted in the IN input parameter. You decide the format of the numeric value by selecting a data type.
- You query the result of the conversion in the OUT output parameter.

Permitted characters for the conversion are the digits "0" to "9", the decimal point, the decimal comma, notations "E" and "e", and the plus and minus characters. The conversion can be interrupted by invalid characters.

Parameters

The following table shows the parameters of the "VAL_STRG" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	USINT, SINT, UINT, INT, UDINT, DINT, REAL, LREAL	I, Q, M, D, L or constant	Value to be converted
SIZE	Input	USINT	I, Q, M, D, L or constant	Number of character positions
PREC	Input	USINT	I, Q, M, D, L or constant	Number of decimal places
FORMAT	Input	WORD	I, Q, M, D, L or constant	Output format of the characters
P	Input	UINT	I, Q, M, D, L	Character starting at which the result is written.
OUT	Output	STRING	D, L	Result of the conversion

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter P

With the P parameter, you specify the character in the string starting at which the result is written. If, for example, the value "2" is specified in the P parameter, the converted value is saved starting at the second character of the string.

Parameter SIZE and P

Use the SIZE parameter to specify how many characters of the character string will be written. This is counted starting from the character specified in the P parameter. If the output value is shorter than the specified length, the result is written to the character string right-justified. The empty character positions are filled with blanks.

Parameter FORMAT

Use the FORMAT parameter to specify how the numerical value is interpreted during conversion and written to the character string. You can specify only tags of the FORMAT data type in the USINT parameter.

The following table shows the possible values of the FORMAT parameter and their meaning:

Value (W#16#....)	Notation	Sign	Decimal representation
0000	Decimal fraction	"-"	"."
0001			","
0002	Exponential		"."
0003			","
0004	Decimal fraction	"+" and "-"	"."
0005			","
0006	Exponential		"."
0007			","
0008 to FFFF	Invalid values		

Parameter PREC

Use the PREC parameter to define the number of decimal places when converting floating-point numbers. A maximum precision of seven numbers is supported for numerical values of the REAL data type. If the value to be converted is an integer, you use the PREC parameter to specify the position where the decimal point will be placed.

Example

The following table shows examples of the conversion of numeric values to a character string.

IN (value)	IN (data type)	P	SIZE	FORMAT (W#16#....)	PREC	OUT (STRING)	ENO status
123	UINT	16	10	0000	0	`xxxxxxx123 C`	1
0	UINT	16	10	0000	2	`xxxxxx0.00 C`	1
12345678	UDINT	16	10	0000	3	`x12345.678 C`	1
12345678	UDINT	16	10	0001	3	`x12345.678 C`	1
123	INT	16	10	0004	0	`xxxxxx+123 C`	1
-123	INT	16	10	0004	0	`xxxxxx-123 C`	1
-0.00123	REAL	16	10	0004	4	`xxx-0.0012 C`	1
-0.00123	REAL	16	10	0006	4	`-1.2300E-3 C`	1
-Inf [1]	REAL	16	10	-/-	4	`xxxxxx-INF C`	0
+Inf [2]	REAL	16	10	-/-	4	`xxxxxx+INF C`	0
NaN [3]	REAL	16	10	-/-	4	`xxxxxxxNaN C`	0
12345678	UDINT	16	6	-/-	3	`xxxxxxxxxx C`	0

"x" represents blanks

[1] -Inf: Floating-point number representing a negative infinite value.

[2] +Inf: Floating-point number representing a positive infinite value.

[3] NaN: Value returned as the result of invalid math operations.

Strg_TO_Chars: Convert character string to Array of CHAR

Description

Use this instruction to copy characters from a character string STRING to a field of several characters of the data type CHAR or BYTE (Array of CHAR / BYTE).

- Specify the character string from which characters are to be copied at the input parameter STRG.
- The characters are written to a data type Array of CHAR or Array of BYTE at the parameter CHARS. With the PCHARS parameter, you specify the position starting at which the characters are to be written to the field Array of CHAR / BYTE. The lower limit of the array is used as standard (example: "1" with Array[1 .. 10] of CHAR).
- The number of characters in the field Array of CHAR must be at least as many characters as are to be copied from the character string STRING.

Only ASCII characters are valid for data types STRING, BYTE and CHAR.

Parameter

The following table shows the parameters of the instruction "Strg_TO_Chars":

Parameter	Declaration	Data type	Memory area	Description
STRG	Input	STRING	D, L	Source: Character string
PCHARS	Input	DINT	I, Q, M, D, L or constant	Position in the destination character string from which the characters will be written.
CHARS	InOut	VARIANT	D, L	Destination: Field in which the characters will be copied. The characters are copied to a field of data type Array of CHAR or Array of BYTE.
CNT	Output	UINT	I, Q, M, D, L	Number of copied characters.

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Chars_TO_Strg: Convert Array of CHAR to character string

Description

Use this instruction to copy characters from a field of several characters of data type CHAR or BYTE (Array of CHAR / BYTE) to a character string STRING.

- Specify the characters of the field Array of CHAR / BYTE to be copied to a character string at the input parameter CHARS. Use the PCHARS parameter to specify the position starting at which the characters of the Array are to be copied. The lower limit of the array is used as standard (example: "1" with Array[1 .. 10] of CHAR).
- The characters are written to a data type STRING at the parameter STRG. The number of characters in the character string STRING must be at least as many characters as are to be copied from the field Array of CHAR.

Only ASCII characters are valid for data types STRING, CHAR and BYTE.

Parameter

The following table shows the parameters of the instruction "Chars_TO_Strg":

Parameter	Declaration	Data type	Memory area	Description
CHARS	Input	VARIANT	I, Q, M, D, L	Source: Field from which the characters will be copied.
PCHARS	Input	DINT	I, Q, M, D, L or constant	Position in the field Array of CHAR / Array of BYTE from which the characters will be copied.
CNT	Input	UINT	I, Q, M, D, L or constant	Number of characters to be copied. Use "0" to copy all characters.
STRG	Output	STRING	D, L	Destination: Character string

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

ATH: Convert ASCII string to hexadecimal number

Description

You use this instruction to convert the ASCII character string specified in the IN input parameter to a hexadecimal number. The result of the conversion is output to the OUT output parameter.

- You can reference the following data types using the pointer in the IN parameter (ASCII): STRING, Array of CHAR, Array of BYTE.
- You can reference the following data types using the pointer in the OUT parameter (hexadecimal): Bit strings, integers, STRING, Array of CHAR, Array of BYTE.

With the N parameter, you specify the number of ASCII characters to be converted. A maximum of 32 767 valid ASCII characters can be converted. Only digits "0" to "9", upper case letters "A" to "F", and lower case letters "a" to "f" can be interpreted. All other characters are converted to zeros.

Since 8 bits are required for the ASCII character and only 4 bits for the hexadecimal digit, the output word length is only half of the input word length. The ASCII characters are converted and positioned in the output in the same order as they are read in. If there is an odd number of ASCII characters, the hexadecimal number is padded with zeros in the nibble to the right of the last converted hexadecimal number.

Parameters

The following table shows the parameters of the "ATH" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	VARIANT	I, Q, M, D, L or constant	Pointer to ASCII character string
N	Input	INT	I, Q, M, D, L or constant	Number of ASCII characters to be converted
RET_VAL	Return	WORD	I, Q, M, D, L	Status of the instruction
OUT	Output	VARIANT	I, Q, M, D, L	Hexadecimal number

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Error code (W#16#....)	Description
0000	No error
0007	Invalid character. Only the following ASCII characters may be used: Digits "0" to "9", upper case letters "A" to "F", lower case letters "a" to "f".
8101	Invalid pointer in the IN parameter, e.g., because a non-existing data block is referenced.
8182	Input buffer is too small for data in the N parameter.
8120	Invalid format in the IN parameter.
8151	Non-supported data type in the IN parameter.
8301	Invalid pointer in the OUT parameter, e.g., because a non-existing data block is referenced.
8382	Output buffer is too small for data in the N parameter.
8320	Invalid format in the OUT parameter.
8351	Non-supported data type in the OUT parameter.

ASCII characters and hexadecimal values

The following table shows the ASCII characters and the corresponding hexadecimal values:

ASCII character	ASCII-coded hexadecimal value	Hexadecimal digit
"0"	30	0
"1"	31	1
"2"	32	2
"3"	33	3
"4"	34	4
"5"	35	5
"6"	36	6
"7"	37	7
"8"	38	8
"9"	39	9
"A"	41	A
"B"	42	B
"C"	43	C
D	44	D
E	45	E
F	46	F

Example

The following table shows examples of the conversion of ASCII character strings to hexadecimal numbers:

IN	N	OUT	ENO status
'0123'	4	16#0123	1
'123AFx1a23'	10	16#123AF01a23	0

HTA: Convert hexadecimal number to ASCII string

Description

You use this instruction to convert the hexadecimal number specified at the IN input to an ASCII string. The result of the conversion is stored at the address specified in the OUT parameter.

- You can reference the following data types using the pointer in the IN parameter (hexadecimal): Bit strings, integers, STRING, Array of CHAR, Array of BYTE.
- You can reference the following data types using the pointer in the OUT parameter (ASCII): STRING, Array of CHAR, Array of BYTE.

With the N parameter, you specify the number of hexadecimal bytes to be converted. Since 8 bits are required for the ASCII character and only 4 bits for the hexadecimal digit, the output value is twice as long as the input value. Each nibble of the hexadecimal number is converted to a character while maintaining the original order.

A maximum of 32 767 characters can be written to the ASCII character string. The result of the conversion is represented by the digits "0" to "9" and upper-case letters "A" to "F".

If the complete result of the conversion cannot be displayed in the OUT parameter, the result is will only be partially written to the parameter.

Parameter

The following table shows the parameters of the "HTA" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	VARIANT	I, Q, M, D, L or constant	Start address of the hexadecimal digits
N	Input	INT	I, Q, M, D, L or constant	Number of hexadecimal bytes to be converted
RET_VAL	Return	WORD	I, Q, M, D, L	Error message
OUT	Output	VARIANT	I, Q, M, D, L	Address at which the result is stored.

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Error code (W#16#....)	Description
0000	No error
8101	Invalid pointer in the IN parameter, e.g., because a non-existing data block is referenced.
8182	Input buffer is too small for data in the N parameter.
8120	Invalid format in the IN parameter.
8151	Non-supported data type in the IN parameter.
8301	Invalid pointer in the OUT parameter, e.g., because a non-existing data block is referenced.
8382	Output buffer is too small for data in the N parameter.
8320	Invalid format in the OUT parameter.
8351	Non-supported data type in the OUT parameter.

ASCII characters and hexadecimal values

The following table shows the ASCII characters and the corresponding hexadecimal values:

Hexadecimal digit	ASCII-coded hexadecimal value	ASCII character
0	30	"0"
1	31	"1"
2	32	"2"
3	33	"3"
4	34	"4"
5	35	"5"
6	36	"6"
7	37	"7"
8	38	"8"
9	39	"9"
A	41	"A"
B	42	"B"
C	43	"C"
D	44	"D"
E	45	"E"
F	46	"F"

Example

The following table shows examples of the conversion of hexadecimal numbers to ASCII character strings:

IN	N	OUT	ENO status
W#16#0123	2	'0123'	1
16#123AF01023	4	'123AF010'	0

Other instructions

LEN: Determine the length of a character string

Description

A tag of the STRING data type contains two lengths: the maximum length and the current length (this is the number of currently valid characters). The maximum length of the character string is specified for each tag in the STRING keyword in square brackets. The current length represents the number of the character places actually used. The current length must be less than or equal to the maximum length. The number of bytes occupied by a string is 2 greater than the maximum length.

You use this instruction to query the current length of the character string specified in the IN input parameter and output this information as a numerical value in the OUT output parameter. An empty string ('') has the length zero.

If errors occur during processing of the instruction, then an empty string will be output.

Parameters

The following table shows the parameters of the "LEN" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING	D, L or constant	Character string
OUT	Return	INT, DINT, REAL, LREAL	I, Q, M, D, L	Number of valid characters

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

CONCAT: Combine character strings

Description

You use this instruction to combine the character string in the IN1 input parameter with the character string in the IN2 input parameter. The result is output in the OUT output parameter with the STRING format. If the resulting character string is longer than the tag specified in the OUT output parameter, then the resulting character string will be limited to the available length.

If errors occur during processing of the instruction and the OUT output parameter can be written, an empty string will be output.

Parameters

The following table shows the parameters of the "CONCAT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	STRING	D, L or constant	Character string
IN2	Input	STRING	D, L or constant	Character string
OUT	Return	STRING	D, L	Resulting character string

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

LEFT: Read the left character of a character string

Description

You use this instruction to extract a partial string beginning with the first character of the string in the IN input parameter. You specify the number of characters to be extracted in the L parameter. The extracted characters are output in the OUT output parameter with STRING format.

If the number of characters to be extracted is greater than the current length of the character string, the OUT output parameter returns the input character string as a result. If the L parameter contains the value "0" or the input value is an empty string, an empty string will be returned. If the value in the L parameter is negative, an empty string will be output.

If errors occur during processing of the instruction and the OUT output parameter can be written, an empty string will be output.

Parameters

The following table shows the parameters of the "LEFT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING	D, L or constant	Character string
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Number of characters to be extracted
OUT	Return	STRING	D, L	Extracted partial character string

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

RIGHT: Read the right characters of a character string

Description

You use this instruction to extract the last L character in a character string in the input parameter IN. You specify the number of characters to be extracted in the L parameter. The extracted characters are output in the OUT output parameter with STRING format.

If the number of characters to be extracted is greater than the current length of the character string, the OUT output parameter returns the input character string as a result. If the L parameter contains the value "0" or the input value is an empty string, an empty string will be returned. If the value in the L parameter is negative, an empty string will be output.

If errors occur during processing of the instruction and the OUT output parameter can be written, an empty string will be output.

Parameters

The following table shows the parameters of the "RIGHT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING	D, L or constant	Character string
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Number of characters to be extracted
OUT	Return	STRING	D, L	Extracted partial character string

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

MID: Read middle characters of a character string

Description

You use this instruction to extract a portion of the character string in the IN input parameter. With the P parameter, you specify the position of the first character to be extracted. With the L parameter, you define the length of the character string to be extracted. The extracted partial character string is output to the OUT output parameter.

The following rules must be observed when executing the instruction:

- If the number of characters to be extracted exceeds the current length of the character string in the IN input parameter, a partial character string will be output, starting from character position P and continuing to the end of the character string.
- If the character position specified in the P parameter falls outside the current character string length in the IN input parameter, an empty character string will be output in the OUT output parameter.
- If the value of the P or L parameter equals zero or is negative, an empty character string will be output in the OUT output parameter.

If errors occur during processing of the instruction and the OUT output parameter can be written, an empty string will be output.

Parameters

The following table shows the parameters of the "MID" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING	D, L or constant	Character string
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Length of the string to be extracted
P	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Position of the first character to be extracted (first character = 1)
OUT	Return	STRING	D, L	Extracted partial character string

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

DELETE: Delete characters in a character string

Description

You use this instruction to delete a portion of the character string in the IN input parameter. With the P parameter, you specify the position of the first character to be deleted. You specify the number of characters to be deleted in the L parameter. The remaining partial character string is output to the OUT output parameter with STRING format.

The following rules must be observed when executing the instruction:

- If the value in the P parameter is less than or equals zero, an empty character string will be output in the OUT output parameter.
- If the value in the P parameter is greater than the current length of the character string in the IN input, the input character string will be returned in the OUT output parameter.
- If the value in the L parameter equals zero, the input character string will be returned in the OUT output parameter.
- If the number of characters to be deleted at the L parameter is greater than the length of the character string in the IN parameter, an empty character string will be output.
- If the value in the L parameter is negative, an empty character string will be output.

If errors occur during processing of the instruction and the OUT output parameter can be written, an empty string will be output.

Parameter

The following table shows the parameters of the "DELETE" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING	D, L or constant	Character string
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Number of characters to be deleted
P	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Position of first character to be deleted
OUT	Return	STRING	D, L	Resulting character string

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

INSERT: Insert characters in a character string

Description

You use this instruction to insert the character string in the IN2 input parameter to the character string in the IN1 input parameter. With the P parameter, you specify the position of the character starting at which the characters are inserted. The result is output in the OUT output parameter with the STRING format.

The following rules must be observed when executing the instruction:

- If the value in the P parameter exceeds the current length of the character string in the IN1 input parameter, the character string of the IN2 input parameter will be appended to the character string of the IN1 input parameter.
- If the value at the P parameter is zero, the character string at the IN2 parameter followed by the character string at the IN1 parameter will be output in the OUT output parameter.
- If the value in the P parameter is negative, an empty character string will be output in the OUT output parameter.
- If the resulting character string is longer than the tag specified in the OUT output parameter, the resulting character string will be limited to the available length.

Parameter

The following table shows the parameters of the "INSERT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	STRING	D, L or constant	Character string
IN2	Input	STRING	D, L or constant	String to insert
P	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Insert position
OUT	Return	STRING	D, L	Resulting character string

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

REPLACE: Replace characters in a character string

Description

You use this instruction to replace a portion of the character string in the IN1 input with the character string in the IN2 input. You specify the position of the first character to be replaced in the P parameter. You specify the number of characters to be replaced in the L parameter. The result is output in the OUT output parameter with the STRING format.

The following rules must be observed when executing the instruction:

- If the value in the P parameter is less than or equals zero, an empty character string will be output in the OUT output parameter.
- If the value in the L parameter is less than zero, an empty character string will be output in the OUT output parameter.
- If the value in the P parameter exceeds the current length of the character string in the IN1 input parameter, the content of the character string in the IN1 parameter will be written to the OUT output parameter.
- If P equals one, the character string in the IN1 input will be replaced beginning with (and including) the first character.
- If the value in the P parameter exceeds the current length of the character string in the IN1 input parameter, the character string of the IN2 input parameter will be appended to the character string of the IN1 input parameter.
- If the resulting character string is longer than the tag specified in the OUT output parameter, the resulting character string will be limited to the available length.

Parameters

The following table shows the parameters of the "REPLACE" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	STRING	D, L or constant	String with characters to be replaced.
IN2	Input	STRING	D, L or constant	String with characters to be inserted.
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Number of characters to be replaced
P	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Position of first character to be replaced
OUT	Return	STRING	D, L	Resulting character string

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

FIND: Find characters in a character string

Description

You can use this instruction to search through the character string in the IN1 input parameter for a specific character or a specific string of characters.

- You specify the value to be searched for in the IN2 input parameter. The search is made from left to right.
- The position of the first occurrence is output in the OUT output parameter. If the search returns no match, the value "0" will be output in the OUT output parameter.

If errors occur during processing of the instruction, an empty string will be output.

Parameters

The following table shows the parameters of the "FIND" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	STRING	D, L or constant	String searched through
IN2	Input	STRING, CHAR	D, L or constant (For CHAR also I, Q, M)	Characters to search for
OUT	Return	DINT, INT, LREAL, REAL	I, Q, M, D, L	Character position

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

9.8.3.3 Distributed I/O

RDREC: Read data record

Description

You use the instruction to read the data record with the number INDEX from the component addressed using the ID. This may be a module in a central rack or a distributed component (PROFIBUS DP or PROFINET IO).

Use MLEN to specify the maximum number of bytes you want to read. The selected length of the destination area RECORD should have at least the length of MLEN bytes.

The value TRUE for the output parameter VALID indicates that the data record was successfully transferred to the destination area RECORD. In this case, the LEN output parameter contains the length of the read data in bytes.

If an error has occurred during transfer of the data record, then this fact will be indicated by the output parameter ERROR. In this case, the output parameter STATUS contains the error information.

Note

If a DPV1 slave is configured via GSD file (GSD rev. 3 and higher) and the DP interface of the DP master is set to "S7 compatible", then you may not read any data records from the I/O modules in the user program with "RDREC". In this case, the DP master addresses the wrong slot (configured slot + 3).

Remedy: set the interface of the DP master to "DPV1".

Note

The interface of the "RDREC" instruction is identical to the "RDREC" FB defined in "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Functional description

"RDREC" works asynchronously, that is, its execution extends over multiple calls. You start the data record transfer by calling "RDREC" with REQ = 1.

The job status is displayed via output parameter BUSY and the two central bytes of output parameter STATUS. The two central bytes of STATUS correspond to the RET_VAL output parameter of the instructions that operate asynchronously.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 1220).

The transfer of the data record is complete when the output parameter BUSY has the value FALSE .

Parameter

The following table shows the parameters of the instruction "RDREC":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Transfer data record
ID	Input	HW_IO (WORD)	I, Q, M, L or constant	Logical address of the DP slave/PROFINET IO component (module or sub-module) • For an output module, bit 15 must be set. • For a mixed module, the lower of the two addresses must be specified.
INDEX	Input	BYTE, DINT, INT, SINT, UINT, USINT, WORD	I, Q, M, D, L or constant	Data record number
MLEN	Input	BYTE, UINT, USINT	I, Q, M, D, L or constant	maximum length in bytes of the data record information to be read
VALID	Output	BOOL	I, Q, M, D, L	New data record was received and is valid
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during the reading process.
STATUS	Output	DWORD	I, Q, M, D, L	Block status or error information
LEN	Output	UINT	I, Q, M, D, L	Length of the read data record information
RECORD	InOut	VARIANT	I, Q, M, D, L	Destination area for the read data record

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Note

If you use "RDREC" to read a data record for PROFINET IO, then negative values in the INDEX, MLEN, and LEN parameters will be interpreted as an unsigned 16-bit integer.

Parameter STATUS

For interpretation of the STATUSparameter, see Parameter STATUS (Page 1706).

WRREC: Write data record

Description

You use this instruction to transfer the RECORD data record to the component addressed using ID. This may be a module in a central rack or a distributed component (PROFIBUS DP or PROFINET IO).

Use LEN to specify the length of the data record to be transmitted in bytes. The selected length of the source area RECORD should have at least the length of LEN bytes.

The value TRUE at output parameter DONE indicates that the data record has been successfully transferred.

If an error has occurred during transfer of the data record, this is indicated by the output parameter ERROR. In this case, the output parameter STATUS contains the error information.

Note

If a DPV1 slave is configured via a GSD file (GSD Rev. 3 and higher) and the DP interface of the DP master is set to "S7 compatible", data records must not be written to the I/O modules in the user program with WRREC. In this case, the DP master addresses the wrong slot (configured slot + 3).

Remedy: set the interface of the DP master to "DPV1".

Note

The interface of the "WRREC" instruction is identical to the "WRREC" FB defined in "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Functional description

"WRREC" works asynchronously, that is, its execution extends over multiple calls. You start the data record transfer by calling WRREC with REQ = 1.

The job status is displayed via the output parameter BUSY and the two central bytes of output parameter STATUS. The two central bytes of STATUS correspond to the RET_VAL output parameter of the instructions that operate asynchronously.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 1220).

Note that you must assign the same value to the actual parameter of RECORD for all "WRREC" calls that belong to one and the same job. The same applies to the actual parameters of LEN.

The transfer of the data record is complete when the output parameter BUSY has the value FALSE.

Parameter

The following table shows the parameters of the instruction "WRREC":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ= 1: Transfer data record
ID	Input	HW_IO (WORD)	I, Q, M, L or constant	Logical address of the DP slave/PROFINET IO component (module or sub-module). For an output module, bit 15 must be set (for example, for address 5: ID:=DW#16#8005). For a mixed module, the lower of the two addresses must be specified. The logical address is converted automatically to DWORD data type.
INDEX	Input	BYTE, DINT, INT, SINT, UINT, USINT, WORD	I, Q, M, D, L or constant	Data record number
LEN	Input	BYTE, UINT, USINT	I, Q, M, D, L or constant	(hidden) Maximum length of the data record to be transferred in bytes
DONE	Output	BOOL	I, Q, M, D, L	Data record was transferred
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during the writing process.
STATUS	Output	DWORD	I, Q, M, D, L	Block status or error information For interpretation of the STATUSparameter, see Parameter STATUS (Page 1706).
RECORD	InOut	VARIANT	I, Q, M, D, L	Data record

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Note

If you use "WRREC" to write a data record for PROFINET IO, negative values in the INDEX and LEN parameters will be interpreted as an unsigned 16-bit integer.

Parameter STATUS

For interpretation of the STATUSparameter, see Parameter STATUS (Page 1706).

RALRM: Receive interrupt

Description RALRM

Description

The instruction receives an interrupt with all corresponding information from an I/O module (centralized structure) or from a DP slave or PROFINET IO device component; it supplies this information to its output parameters.

The information in the output parameters contains the start information of the called OB as well as information of the interrupt source.

Call "RALRM" only within the interrupt OB started by the CPU operating system as a result of the I/O interrupt that is to be examined.

Note

If you call "RALRM" in an OB whose start event is not an I/O interrupt, the instruction will provide correspondingly reduced information in its outputs.
Make sure to use different instance DBs when you call "RALRM" in different OBs. If you evaluate data resulting from an "RALRM" call outside of the associated interrupt OB, you should moreover use a separate instance DB per OB start event.

Note

The interface of the "RALRM" instruction is identical to the "RALRM" FB defined in "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Calling RALRM

"RALRM" can be called in three operating modes (MODE parameter). These are explained in the table below.

MODE	RALRM ...
0	... shows the component that triggered the interrupt in the ID output parameter and writes TRUE in the NEW output parameter.
1	... writes all output parameters, independent of the interrupt triggering component.
2	... checks whether the component specified in the F_ID input parameter has triggered the interrupt. • If not, NEW = FALSE • If yes, NEW = TRUE and all other output parameters are written.

Parameter

The following table shows the parameters of the instruction "RALRM":

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Mode
F_ID	Input	HW_IO (WORD)	D, L or constant	Logical start address of the component (module) from which interrupts will be received
MLEN	Input	BYTE, UINT, USINT	I, Q, M, D, L or constant	Maximum length of the interrupt information to be received, in bytes
NEW	Output	BOOL	I, Q, M, D, L	A new interrupt was received.
STATUS (Page 1706)	Output	DWORD	I, Q, M, D, L	Error code of the instruction or DP master
ID	Output	HW_IO (WORD)	I, Q, M, L or constant	Logical start address of the component (module) from which an interrupt was received Bit 15 contains the I/O ID: 0 = input address, 1 = output address.
LEN	Output	DINT, DWORD, LREAL, REAL, UDINT, UINT	I, Q, M, D, L	Length of the received interrupt information
TINFO (Page 1709)	InOut	VARIANT	I, Q, M, D, L	Destination area for OB start and management information
AINFO (Page 1713)	InOut	VARIANT	I, Q, M, D, L	Destination area for header information and additional interrupt information For AINFO , you should provide a length of at least MLEN bytes.

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Note

If you select a destination area (Page 1720)TINFO or AINFO that is too short, RALRM cannot enter the full information.

Parameter STATUS

Description

The STATUS output parameter contains error information. If it is interpreted as ARRAY[1...4] of BYTE, then error information will have the following structure:

Array element	Name	Meaning
STATUS[1]	Function_Num	• B#16#00, if no error • Function ID from DPV1-PDU: In the event of an error, B#16#80 is output (in the event of an error reading a data record B#16#DE and writing a data record B#16#DF). If no DPV1 protocol element is used, then B#16#C0 will be output.
STATUS[2]	Error_Decode	Location of the error ID
STATUS[3]	Error_Code_1	Error ID
STATUS[4]	Error_Code_2	Manufacturer-specific error ID expansion

Array element STATUS[2]

STATUS[2] can have the following values:

Error_Decode (B#16#...)	Source	Meaning
00 to 7F	CPU	No error or no warning
80	DPV1	Error according to IEC 61158-6
81 to 8F	CPU	B#16#8x shows an error in the xth call parameter of the instruction.
FE, FF	DP profile	Profile-specific error

Array element STATUS[3]

STATUS[3] can have the following values:

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
00	00		No error, no warning
70	00	reserved, reject	Initial call; no active data record transfer
	01	reserved, reject	Initial call; data record transfer has started
	02	reserved, reject	Intermediate call; data record transfer already active
80	90	reserved, pass	Invalid logical start address
	92	reserved, pass	Illegal type for VARIANT pointer

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
	93	reserved, pass	The DP component addressed via ID or F_ID is not configured.
	96		The "RALRM (Page 1704)" cannot supply the OB start information, management information, header information, or additional interrupt information. For OBs 4x, 55, 56, 57, 82, and 83, you can use the "DPNRM_DG (Page 1725)" instruction to read the current diagnostics message frame of the relevant DP slave asynchronously (address information from OB start information).
	A0	read error	Negative acknowledgment while reading the module.
	A1	write error	Negative acknowledgement when writing to the module
	A2	module failure	DP protocol error at layer 2 (e.g., slave failure or bus problems)
	A3	reserved, pass	• PROFIBUS DP: DP protocol error with Direct-Data-Link-Mapper or User-Interface/User • PROFINET IO: General CM error
	A4	reserved, pass	Communication on the communication bus disrupted
	A5	reserved, pass	–
	A7	reserved, pass	DP slave or module is occupied (temporary error
	A8	version conflict	DP slave or module reports non-compatible versions
	A9	feature not supported	Function is not supported by DP slave or module
	AA to AF	user specific	DP slave or module reports a manufacturer-specific error in its application. Please check the documentation from the manufacturer of the DP slave or module.
	B0	invalid index	Data record not known in module Illegal data record number ≥ 256
	B1	write length error	The length information in the RECORD parameter is incorrect With "RALRM (Page 1704)": length error in AINFO (Page 1713), With "RDREC (Page 1700)" and "WRREC (Page 1702)": length error in MLEN
	B2	invalid slot	The configured slot is not occupied.
	B3	type conflict	Actual module type does not match specified module type
	B4	invalid area	DP slave or module reports access to an invalid area
	B5	state conflict	DP slave or module not ready
	B6	access denied	DP slave or module denies access
	B7	invalid range	DP slave or module reports an invalid range for a parameter or value
	B8	invalid parameter	DP slave or module reports an invalid parameter

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
	B9	invalid type	DP slave or module reports an invalid type With "RDREC (Page 1700)": buffer too small (subsets cannot be read) With "WRREC (Page 1702)": buffer too small (subsets cannot be written)
	BA to BF	user specific	DP slave or module reports a manufacturer-specific error when accessing. Please check the documentation from the manufacturer of the DP slave or module.
	C0	read constrain conflict	With "WRREC (Page 1702)": the data can only be written when the CPU is in STOP mode. Note: this means that writing by the user program is not possible. You can only write the data online with PG/PC. With "RDREC (Page 1700)": the module routes the data record, but either no data is present or the data can only be read when the CPU is in STOP mode. Note: if data can only be read when the CPU is in STOP mode, then an evaluation by the user program is not possible. In this case, you can only read the data online with PG/PC.
	C1	write constrain conflict	The data of the previous write job on the module for the same data record have not yet been processed by the module.
	C2	resource busy	The module is currently processing the maximum possible number of jobs for a CPU.
	C3	resource unavailable	The required operating resources are currently occupied.
	C4		Internal temporary error. Job could not be carried out. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.
	C5		DP slave or module not available.
	C6		Data record transfer was canceled due to priority class cancellation
	C7		Job aborted due to warm or cold restart on the DP master
	C8 to CF		DP slave or module reports a manufacturer-specific resource error. Please check the documentation from the manufacturer of the DP slave or module.
	Dx	user specific	DP slave specific. Refer to the description of the DP slave.
81	00 to FF		Error in the initial call parameter (with "RALRM (Page 1704)": MODE)
	00		Illegal operating mode
82	00 to FF		Error in the second call parameter
:	:		:
88	00 to FF		Error in the eighth call parameter (with "RALRM (Page 1704)": TINFO (Page 1709))
	01		Wrong syntax ID
	23		Quantity structure exceeded or destination area too small

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
	24		Wrong range ID
	32		DB/DI no. out of user range
	3A		DB/DI no. is NULL for area ID DB/DI or specified DB/DI does not exist
89	00 to FF		Error in the ninth call parameter (with "RALRM (Page 1704)": AINFO (Page 1713))
	01		Wrong syntax ID
	23		Quantity structure exceeded or destination area too small
	24		Wrong range ID
	32		DB/DI no. out of user range
	3A		DB/DI no. is NULL for area ID DB/DI or specified DB/DI does not exist
8A	00 to FF		Error in the 10th call parameter
:	:		:
8F	00 to FF		Error in the 15th call parameter
FE, FF	00 to FF		Profile-specific error

Array element STATUS[4]

With DPV1 errors, the DP master passes on STATUS[4] to the CPU and the instruction. Without DPV1 error, this value is set to 0, with the following exceptions for "RDREC":

- STATUS[4] contains the destination area length from RECORD, if MLEN > the destination area length from RECORD
- STATUS[4]=MLEN, if the actual data record length < MLEN < the destination area length from RECORD
- STATUS[4]=0, if STATUS[4]> 255 would have to be set

In PROFINET IO, STATUS[4] has the value "0".

Parameter TINFO

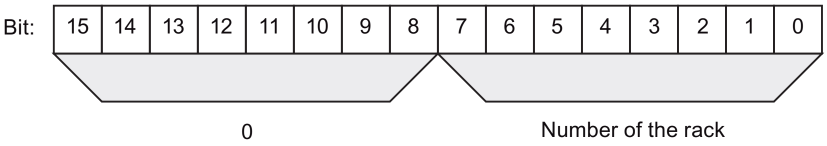
Data structure of the destination area TINFO

Byte	Meaning
0 to 19	Start information of the OB in which "RALRM (Page 1704)" was currently called
20 and 21	Address, for exact description, see below
22 to 31	Management information, for exact description, see below

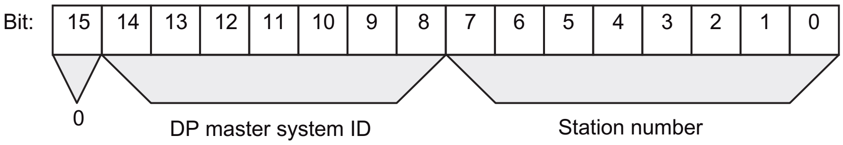
Structure of the address (bytes 20 and 21)

The address contains:

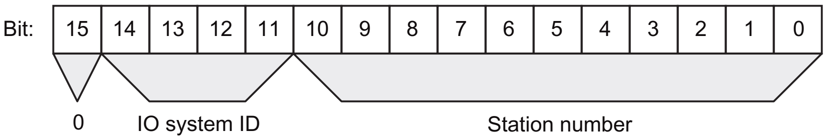
- In a central configuration, the rack number (0-31).



- In a distributed configuration with PROFIBUS DP
  - The DP master system ID (1-31)
  - The station number (0-127).



- In a distributed configuration with PROFINET IO:
  - The last two positions in the PROFINET IO system ID (0-15). To obtain the complete PROFINET IO system ID, you must add 100 (decimal) to it.
  - The station number (0-2047).



Structure of the management information in bytes 20 to 25

Byte no. for TINFO	Data type	Meaning				
20	BYTE	central:	0			
		distributed:	PROFIBUS DP: DP master ID: 1 to 31)			
			PROFINET IO: see above			
21	BYTE	central:	Module rack number (possible values: 0 to 31)			
		distributed:	Number of the DP station (possible values: 0 to 127)			
			PROFINET IO: see above			
22	BYTE	central:	0: Data record 0 or data record 1			
		distributed:	Bit 0 to 3:	Slave type	0000: 0001: 0010: 0011: 0100 – 0111: 1000: 1001 and higher:	DP (data record 0 structure) DPS7 (data record 0 or data record 1 structure) DPS7 V1 (data record 0 or data record 1 structure) DPV1 (structure acc. to PROFIBUS DP standard) reserved PROFINET IO (structure acc. to PROFINET IO Standard) reserved
			Bit 4 to 7:	Profile type		Reserved
23	BYTE	central:	0			
		distributed:	Bit 0 to 3:	Alarm info type	0000:	Transparent, which is always the case for PROFINET IO (interrupt originates from a configured distributed module)
					0001:	Representative (interrupt originates from a non-DPV1 slave/non IO device or a slot that is not configured)
					0010:	Generated (interrupt generated in the CPU)
					0011 and higher:	Reserved
			Bit 4 to 7:	Structure version	0000: 0001 and higher:	Initial Reserved
24	BYTE	central:	0			
		distributed:	Flags of the PROFIBUS DP master interface module/PROFINET IO controller master interface module			
			Bit 0 = 0:	Interrupt originating from an integrated interface module (PROFINET IO or PROFIBUS DP)		
			Bit 0 = 1:	Interrupt originating from an external interface module (PROFINET IO or PROFIBUS DP)		
			Bit 1 to 7:	Reserved		

Byte no. for TINFO	Data type	Meaning		
25	BYTE	central:	0	
		distributed:	Flags of the PROFIBUS DP slave interface module	
			Bit 0:	EXT_DIAG_FLAG from the diagnostics message frame, or 0 if this bit does not exist in the interrupt The bit is 1 if the DP slave is faulty.
			Bit 1 to 7:	Reserved
			Flags of the PROFINET IO controller interface module	
			Bit 0:	ARDiagnosisState or 0 if there is no information in the interrupt. The bit is 1 if the IO device is faulty.
			Bit 1 to 7:	Reserved

Structure of the management information in bytes 26 to 27 with PROFIBUS and a central configuration

Byte no. for TINFO	Data type	Meaning	
26 and 27	WORD	central:	0
	WORD	distributed:	PROFIBUS ID number as unique identifier of the PROFIBUS DP slave
28 and 29	WORD	0	(Bytes 28 and 29 can be omitted)
30 and 31	WORD	0	(Bytes 30 and 31 can be omitted)

Structure of the management information in bytes 26 to 31 with PROFINET IO

Byte no. for TINFO	Data type	Meaning	
26 and 27	WORD	distributed:	PROFINET IO device ID number as unique identifier of the PROFINET IO device
28 and 29	WORD	distributed:	Manufacturer ID
30 and 31	WORD	distributed:	ID number of the instance

Parameter AINFO

Data structure of the destination area AINFO with interrupts from PROFIBUS DP or central I/O devices

The information for PROFINET IO is provided below.

Byte	Meaning	
0 to 3	Header information, for exact description, see below	
4 to 199	Additional interrupt information: data for the respective interrupt:	
	central:	ARRAY[0] to ARRAY[195]
	distributed:	ARRAY[0] to ARRAY[59]

Structure of the header information with interrupts from PROFIBUS DP or central IO devices

Byte	Data type	Meaning		
0	BYTE	Length of the received interrupt information in bytes		
		central: distributed:	4 to 224 4 to 63	
1	BYTE	central:	Reserved	
		distributed:	ID for the interrupt type	
			1: 2: 3: 4: 5: 6: 31	Diagnostics interrupt Process interrupt Removal interrupt Insertion interrupt Status interrupt Update Interrupt Failure of an expansion device, DP master system, or DP station
			32 to 126:	Manufacturer-specific interrupt
2	BYTE	Slot number of the component that triggered the interrupt		
3	BYTE	central:	Reserved	
		distributed:	Specifier	
			Bits 0 and 1:	0: no further information; 1: Incoming event, faulty slot 2: Outgoing event, slot not faulty anymore 3: Outgoing event, slot still faulty
			Bit 2:	Add_Ack
			Bits 3 to 7:	Sequence number

Data structure of the destination area AINFO with interrupts from PROFINET IO

Byte	Meaning
0 to 25	Header information, for exact description, see below
26 to 1431	Additional interrupt information: Standardized diagnostics data for the respective interrupt: ARRAY[0] to ARRAY[1405] Note: The additional interrupt information may also be omitted.

Structure of the header information with interrupts from PROFINET IO

Byte	Data type	Meaning
0 and 1	WORD	• Bits 0 to 7: Block type • Bits 8 to 15: Reserved
2 and 3	WORD	Block length
4 and 5	WORD	Version: • Bits 0 to 7: low byte • Bits 8 to 15: high byte
6 and 7	WORD	ID for interrupt type: • 1: Diagnostics interrupt (incoming) • 2: Process interrupt • 3: Remove module interrupt • 4: Insert module interrupt • 5: Status interrupt • 6: Update interrupt • 7: Redundancy interrupt • 8: Controlled by supervisor • 9: Released by supervisor • 10: Configured module not inserted • 11: Return of the sub-module • 12: Diagnostics interrupt (outgoing) • 13: Slave-to-slave connection alarm • 14: Neighborhood change alarm • 15: Clock synchronization message (bus end) • 16: Clock synchronization alarm (device end) • 17: Network component alarm • 18: Time synchronization alarm (bus end) • 19 to 31: Reserved • 32 to 127: Manufacturer-specific interrupt • 128 to 65535: Reserved
8 to 11	DWORD	API (Application Process Identifier)
12 to 13	WORD	Slot number of the component triggering the interrupt (range of values 0 to 65535)

Byte	Data type	Meaning
14 to 15	WORD	Submodule slot number of the component triggering the interrupt (range of values 0 to 65535)
16 to 19	DWORD	Module identification; specific information on the source of the interrupt
20 to 23	DWORD	Submodule identification; specific information on the source of the interrupt
24 to 25	WORD	Interrupt specifier: • Bits 0 to 10: Sequence number (range of values 0 to 2047) • Bit 11: Channel diagnostics: 0: No channel diagnostics available 1: Channel diagnostics information exists • Bit 12: Status of manufacturer-specific diagnostics: 0: No manufacturer-specific status information available 1: Manufacturer-specific status information available • Bit 13: Status of diagnostics for sub-module: 0: No status information available, all errors have been corrected 1: At least one item of channel diagnostics and/or status information is available • Bit 14: Reserved • Bit 15: Application relationship diagnosis state: – 0: None of the modules configured within this application relationship reports diagnostics information – 1: At least one of the modules configured in this AR is reporting diagnostics information

Structure of additional interrupt information with interrupts from PROFINET IO

The additional interrupt information for PROFINET IO depends on the format identifier. It can comprise multiple data blocks with the same or different format identifier. The following format identifiers are available:

- W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics

Byte	Data type	Meaning
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics
2 to n	BYTE	See manufacturer's manual.

- W#16#8000: Channel diagnostics

Channel diagnostics is output in blocks of 6 bytes each. The additional interrupt information (without format identifier) is only output for disrupted channels.

Byte	Data type	Meaning	
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#8000: Channel diagnostics	
2 to 3	WORD	Channel number of the component triggering the interrupt (range of values: 0 to 65535): • W#16#0000 to W#16#7FFF: Channel number of the interface module/sub-module • W#16#8000: The generic substitute for the entire sub-module • W#16#8001 to W#16#FFFF: Reserved	
4	BYTE	Bits 0 to 2:	Reserved
		Bits 3 to 4:	Type of error: • 0: Reserved • 1: Incoming error • 2: Outgoing error • 3: Outgoing error, other errors present
		Bits 5 to 7:	Type of channel: • 0: Reserved • 1: Input channel • 2: Output channel • 3: Input/output channel

Byte	Data type	Meaning
5	BYTE	Data format: • B#16#00: Free data format • B#16#01: Bit • B#16#02: 2 bits • B#16#03: 4 bits • B#16#04: Byte • B#16#05: Word • B#16#06: Double word • B#16#07: 2 double words • B#16#08 to B#16#FF: Reserved
6 to 7	WORD	Type of error: • W#16#0000: Reserved • W#16#0001: Short circuit • W#16#0002: Undervoltage • W#16#0003: Overvoltage • W#16#0004: Overload • W#16#0005: Overtemperature • W#16#0006: Wire break • W#16#0007: High limit exceeded • W#16#0008: Low limit exceeded • W#16#0009: Error • W#16#000A to W#16#000F: Reserved • W#16#0010 to W#16#001F: Manufacturer-specific • W#16#0020 to W#16#00FF: Reserved • W#16#0100 to w#16#7FFF: Manufacturer-specific • W#16#8000: Device diagnostics available • W#16#8001 to W#16#FFFF: Reserved Not all channels support every error type. For detailed information, refer to the description of the diagnostics data for the specific device.

Note

The section from "channel number" to "type of error" can occur from 0 to n times.

W#16#8001

W#16#8001: MULTIPLE (different types of diagnostics information are transmitted)

In this case, the additional interrupt information is transmitted as blocks of variable length.

Byte	Data type	Meaning
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#8001: Manufacturer-specific diagnostics and/or channel diagnostics
2 to 3	WORD	Block type
4 to 5	WORD	Block length
6	BYTE	Version: high byte
7	BYTE	Version: low byte
8 to 11	DWORD	API (only if low byte of version = 1)
12 to 13	WORD	Slot number
14 to 15	WORD	Subslot number
16 to 17	WORD	Channel number
18 to 19	WORD	Channel properties
20 to 21	WORD	Format identifier: • W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics • W#16#8000: Channel diagnostics • W#16#8002: Extended channel diagnostics • W#16#8003: Stepped extended channel diagnostics • W#16#8004 to W#16#80FF: Reserved
22 to n	BYTE	Data depend on the format identifier

Note

The section starting from "block type" can occur from 1 to n times.

W#16#8002

W#16#8002: Extended channel diagnostics

Byte	Meaning
0 to 1	Format identifier W#16#8002
2 to 3	Channel number
4 to 5	Channel properties
6 to 7	Error type
8 to 9	Additional error value
10 to 13	Additional error information

W#16#8003

W#16#8003: Stepped extended channel diagnostics

Byte	Meaning
0 to 1	Format identifier W#16#8003
2 to 3	Channel number
4 to 5	Channel properties
6 to 7	Error type
8 to 9	Additional error value
10 to 13	Additional error information
14 to 17	Qualified channel qualifier

W#16#8100

W#16#8100: Maintenance information

Byte	Meaning
0 to 1	Format identifier W#16#8100
2 to 3	Block type
4 to 5	Block length
6 to 7	Block version
8 to 9	Reserved
10 to 13	Maintenance status

Note

You can find more detailed information about the structure of the additional alarm information in the *Programming Manual SIMATIC PROFINET IO from PROFIBUS DP to PROFINET IO* and the current version of IEC 61158-6-10-1.

Destination area TINFO and AINFO

Destination area TINFO and AINFO

Depending on the respective OB in which "RALRM (Page 1704)" is called, the destination areas TINFO and AINFO are only partially written. Refer to the table below to find out which information is entered respectively.

Interrupt type	OB	TINFO OB status information	TINFO Management information	AINFO Header information	AINFO Additional interrupt information	
Process interrupt	4x	Yes	Yes	Yes	central:	No
					distributed:	as supplied by PROFIBUS DP slave/PROFINET IO device
Status interrupt	55	Yes	Yes	Yes	Yes	Yes
Update interrupt	56	Yes	Yes	Yes	Yes	Yes
Manufacturer-specific interrupt	57	Yes	Yes	Yes	Yes	Yes
I/O redundancy error	70	Yes	Yes	No	No	No
Diagnostics interrupt	82	Yes	Yes	Yes	central:	Data record 1
					distributed:	as supplied by PROFIBUS DP slave/PROFINET IO device
Insert/remove interrupt	83	Yes	Yes	Yes	central:	No
					distributed:	as supplied by PROFIBUS DP slave/PROFINET IO device
Special form of the remove module interrupt: Controlled by supervisor	83	Yes	Yes	Yes	PROFINET IO only	
Special form of the insert module interrupt: Enabled by supervisor	83	Yes	Yes	Yes	PROFINET IO only	
Unconfigured module inserted	83	Yes	Yes	Yes	PROFINET IO only	
Rack failure/ station failure	86	Yes	Yes	No	No	
... all other OBs		Yes	No	No	No	

Others

DPRD_DAT: Read consistent data of a DP standard slave

Description

You require "DPRD_DAT" because you can only read out a maximum of four continuous bytes using the load commands that access the I/O or the process image input table.

Note

You can also read in consistent data via the process image input, if applicable.
Refer to the related documentation to find out if your CPU supports this functionality.

You use the "DPRD_DAT" instruction to read out consistent data of a DP standard slave/PROFINET IO device.

For additional information on consistent data of a DP standard slave/PROFINET IO device, refer to Section "Data consistency (Page 1850)".

See the documentation supplied with your CPU for the maximum data length. If there was no error during the data transmission, the data that have been read are entered in the target area indicated by RECORD .

The target area must have at least the same length that you configured for the selected module. If you read from a DP standard slave with a modular configuration or with several DP identifiers, you can only access the data of one module/DP identifier at the configured hardware identifier per "DPRD_DAT" call.

Note

An access with "DPRD_DAT" is limited to data areas with a length of 3 bytes or more than 4 bytes. Otherwise an access will be rejected with error code W#16#8090 .

Parameters

The following table shows the parameters of the instruction "DPRD_DAT":

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO (WORD)	I, Q, M, L or constant	Hardware identifier of the module which is to be read from. If you click on the block parameter LADDR, all addressable components, including hardware identifier are available in a drop-down list.
RET_VAL	Return	DINT, INT, LREAL, REAL	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
RECORD	Output	VARIANT	I, Q, M, D, L	Target area for the user data that were read. It has to be at least as long as the inputs of the selected module. The BYTE, WORD and Array of BYTE / WORD data types are permitted. The data type STRING is not supported.

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Note

If you access DPV1 slaves, error information of these slaves can be forwarded from the DP master to the instruction. For a description of this error information, refer to Parameter STATUS (Page 1706)STATUS[3].

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	• You have not configured a module for the specified hardware identifier or • You have ignored the restriction concerning the length of consistent data, or • you have not specified a hardware identifier as an address at parameter LADDR .
8092	A type other than (Array of) BYTE / WORD is specified in the VARIANT reference.
8093	No DP module/PROFINET IO device from which you can read consistent data exists for the hardware identifier specified in LADDR .
80A0	Access error detected while I/O devices were being accessed.
80B0	Slave failure on external DP interface module.
80B1	The length of the specified target area at parameter RECORD is shorter than the configured user data length.
80B2	System error with external DP interface module.

Error code (W#16#...)	Explanation
80B3	System error with external DP interface module.
80C0	The data haven't yet been read by the module.
80C2	System error with external DP interface module.
80Fx	System error with external DP interface module.
87xy	System error with external DP interface module.
808x	System error with external DP interface module.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 1222)

DPWR_DAT: Write consistent data of a DP standard slave

Description

You require "DPWR_DAT" because you can only write a maximum of four continuous bytes using the transfer commands that access the I/O or process image output.

Note

If required, you can also write consistent data via the process image outputs. Refer to the related documentation to find out if your CPU supports this functionality. Do not use both possibilities concurrently when writing consistent data: Either use "DPWR_DAT" or write via the process image output table.

⚠ CAUTION

When using "DPWR_DAT", avoid accessing I/O areas that have process image partitions with OB6x connections (isochronous mode interrupts) assigned to them.

You use the "DPWR_DAT" instruction to consistently transfer the data in RECORD to the addressed DP standard slave/PROFINET IO device, and, if applicable, to the process image (if you have configured the relevant address area of the DP standard slave as a consistent range in a process image).

For additional information on consistent data of a DP standard slave/PROFINET IO device, refer to Section "Data consistency (Page 1850)".

See the documentation supplied with your CPU for the maximum length of data to be transferred. The data is transferred synchronously, that is, the write process is completed when the instruction is completed. The source area should have the same length that you have configured for the selected module. If the source area at parameter RECORD is longer than the outputs of the configured modules, only the data up to the maximum length of the outputs is transferred.

If the DP standard slave has a modular design, you can only access one module of the DP slave.

Note

An access with "DPWR_DAT" is limited to data areas with a length of 3 bytes or more than 4 bytes. Otherwise an access will be rejected with error code W#16#8090 .

Parameters

The following table shows the parameters of the instruction "DPWR_DAT":

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO (WORD)	I, Q, M, L or constant	Configured hardware identifier from the PIQ area of the module to which the data is written. If you click on the block parameter LADDR, all addressable components, including hardware identifier are available in a drop-down list.
RECORD	Input	VARIANT	I, Q, M, D, L	Source area for the user data to be written. It must be at least as long as what you have configured for the selected module. The BYTE, WORD and Array of BYTE / WORD data types are permitted. The data type STRING is not supported.
RET_VAL	Return	DINT, INT, LREAL, REAL	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Note

If you access DPV1 slaves, error information of these slaves can be forwarded from the DP master to the instruction. For a description of this error information, refer to Parameter STATUS (Page 1706)STATUS[3].

Error code (W#16#...)	Explanation
0000	No error occurred.
808x	System error with external DP interface module.

Error code (W#16#...)	Explanation
8090	• You have not configured a module for the specified hardware identifier or • You have ignored the restriction concerning the length of consistent data, or • you have not specified a hardware identifier at parameter LADDR .
8092	A type other than (Array of) BYTE / WORD is specified in the VARIANT reference.
8093	No DP module/PROFINET IO device to which you can write consistent data exists at the logical address specified in LADDR .
80A1	Access error detected while I/O devices were being accessed.
80B0	Slave failure on external DP interface module.
80B1	The length of the specified target area is not identical to the configured user data length.
80B2	System error with external DP interface module.
80B3	System error with external DP interface module.
80C1	The data of the previous write job on the module have not yet been processed by the module.
80C2	System error with external DP interface module.
80Fx	System error with external DP interface module.
85xy	System error with external DP interface module.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 1222)

PROFIBUS

DPNRM_DG: Read diagnostics data from a DP slave

Description

You use the "DPNRM_DG" instruction to read the current diagnostics data of a DP slave in the form specified in EN 50170 Volume 2, PROFIBUS.

Refer to the following table for the basic structure of the slave diagnostics data and to the manuals of the DP slaves for further information.

Byte	Meaning
0	Station status 1
1	Station status 2
2	Station status 3
3	Master station number
4	Vendor ID (high byte)
5	Vendor ID (low byte)
6 ...	Additional slave-specific diagnostic information

The data that has been read is entered in the destination area indicated by RECORD following error-free data transfer. You start the read process by assigning the value "1" to the REQ input parameter when the "DPNRM_DG" instruction is called.

Functional description

The reading process is executed asynchronously, in other words, it can extend over several calls. The RET_VAL and BUSY output parameters indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 1220).

Parameters

The following table shows the parameters of the instruction "DPNRM_DG":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ=1: Read request
LADDR	Input	HW_DP SLAVE (WORD)	D, L or constant	Configured diagnostic address of the DP slave. Note: Address must be specified in hexadecimal form, for example, diagnostic address 1022 means: LADDR:=W#16#3FE.
RET_VAL	Return	DINT, INT, LREAL, REAL	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. If no error has occurred, the length of the data actually transferred is entered in RET_VAL .
RECORD	Output	VARIANT	I, Q, M, D, L	Destination area for the diagnostics data that were read. Only the BYTE data type is permitted. The minimum length of the data record to be read or the destination area is 6. The maximum length of the data record to be sent is 240. Standard slaves can provide more than 240 bytes of diagnostics data up to a maximum of 244 bytes. In this case, the first 240 bytes are transferred to the destination area and the overflow bit is set in the data.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: The reading process is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RECORD

The CPU evaluates the actual length of the read diagnostics data:

If the length specified for RECORD

- is less than the number of data bytes supplied, the data are discarded and a corresponding error code is entered in RET_VAL .
- is greater than or equal to the number of supplied data bytes, the data are accepted in the destination area and the actual length is entered in RET_VAL as a positive value.

Note

You must ensure that the actual parameters of RECORD match in all calls belonging to a job.

A job is uniquely identified by the LADDR input parameter.

Standard slaves with more than 240 bytes of diagnostics data

With standard slaves on which the number of standard diagnostics data is between 241 and 244 bytes, note the following points:

If the length specified for RECORD

- is less than 240 bytes, the data are discarded and a corresponding error code is entered in RET_VAL .
- If the length specified for RECORD is greater than or equal to 240 bytes, the first 240 bytes of the standard diagnostics data are transferred to the destination area and the overflow bit is set in the data.

Parameter RET_VAL

- If an error occurs while the function is being executed, the return value contains an error code.
- If no error occurs during the data transfer, RET_VAL contains the length of the data read in bytes as a positive number.

Note

The amount of data read in a DP slave depends on its diagnostics status.

For the evaluation of the error information of the RET_VAL parameter, refer to the following table.

The general error information of the instructions are described in the following section: Evaluating errors with output parameter RET_VAL (Page 1222).

Error code (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value "0".	-
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".	Distributed I/O
8090	Specified logical start address invalid: There is no assignment in SDB1/SDB2x or there is no start address.	-
8093	This instruction is not permitted for the module selected by means of LADDR and IOID .	-
80A2	• DP protocol error at layer 2 (for example, slave failure or bus problems) • For ET 200S, data record cannot be read in DPV0 mode.	Distributed I/O
80A3	DP protocol error with user interface/user	Distributed I/O
80A4	Communication on the communication bus disrupted	Error occurs between the CPU and the external DP interface module.
80B0	• Instruction not possible for module type. • The module does not recognize the data record. • Data record number 241 not permitted. • With "WR_REC (Page 1702)", data records 0 and 1 are not permitted.	-
80B1	The length specified in parameter RECORD is incorrect.	Specified length < record length
80B2	The configured slot is not occupied.	-
80B3	Actual module type does not match the specified module type in SDB1.	-
80C0	There are no diagnostics data available.	-
80C1	The data of the previous write job on the module for the same data record have not yet been processed by the module.	-
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	-
80C3	The required resources (memory, etc.) are currently occupied.	-
80C4	Internal temporary error. Job could not be carried out. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	-
80C5	Distributed I/O not available.	Distributed I/O
80C6	Data record transfer stopped due to priority class abort (restart or background)	Distributed I/O
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 1222)	-

See also

RDREC: Read data record (Page 1700)

9.8.3.4 Interrupts

ATTACH: Attach an OB to an interrupt event

Description

You use this instruction to assign an organization block (OB) to an event.

You enter the symbolic or numeric name of the organization block in the OB_NR parameter. This will then be assigned to the event specified in the EVENT parameter.

If the event in the EVENT parameter occurs following error-free execution of the "ATTACH" instruction, the organization block in the OB_NR parameter will be called and its program executed.

With the ADD parameter, you specify whether previous assignments of the organization block to other events should be canceled or retained. If the ADD parameter has the value "0", the existing assignments will be replaced by the current assignment.

Parameters

The following table shows the parameters of the "ATTACH" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_ATT (INT)	I, Q, M, D, L or constant	Organization block (numbers up to 32768 are supported.)
EVENT	Input	EVENT_ATT (DWORD)	D, L or constant	Event
ADD	Input	BOOL	I, Q, M, D, L or constant	Effects on previous assignments: • ADD=0 (default): This event replaces all previous event assignments for this OB. • ADD=1: This event is added to the previous event assignments for this OB.
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Error code (W#16#....)	Description
0	No error
8090	OB does not exist
8091	OB is incorrect type
8093	Event does not exist

DETACH: Detach an OB from an interrupt event

Description

You use this instruction to cancel the existing assignment of an organization block to one or more events during runtime.

- If you have selected a single event, the assignment of the OB to this event will be cancelled. All other currently existing assignments remain active. You can select an individual event using the drop-down list of the operand placeholder at the EVENT parameter.
- If you have not selected an event, all currently existing assignments of the organization block to events will be canceled.

You enter the symbolic or numeric name of the organization block in the OB_NR parameter. The assignment of this organization block to the event specified in the EVENT parameter will then be canceled.

Parameters

The following table shows the parameters of the "DETACH" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_ATT (INT)	I, Q, M, D, L or constant	Organization block (numbers up to 32768 are supported.)
EVENT	Input	EVENT_ATT (DWORD)	D, L or constant	Event
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Error code (W#16#....)	Description
0	No error
1	No assignment exists (warning)
8090	OB does not exist
8091	OB has incorrect type
8093	Event does not exist

Cyclic interrupt

SET_CINT: Set cyclic interrupt parameters

Description

You use this instruction to set the parameters for a cyclic interrupt OB. The start time for a cyclic interrupt OB is generated from the respective time interval of the OB and the phase offset.

- The time interval of an OB is the interval at which the OB is periodically called. For example, if the time interval is 100 μs, the OB will be called every 100 μs during program execution.
- The phase offset is a time interval by which the call of a cyclic interrupt OB is offset. You can use the phase offset to process low priority organization blocks in a precise time base.

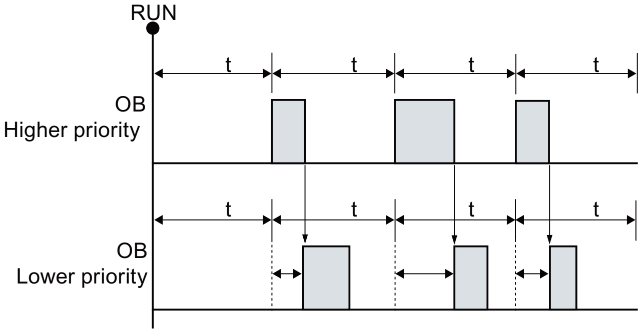
If the OB does not exist or if the time interval used is not supported, a corresponding error alarm is output in the RET_VAL parameter.

A time interval in the CYCLE parameter of "0" means that the OB will not be called.

Functional description

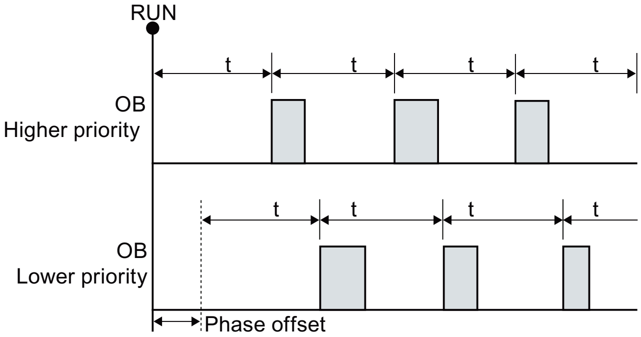
If a lower priority OB and a higher priority OB are called in the same time interval, the lower priority OB will only be called once the higher priority OB has been executed. The call time for the lower priority OB can be offset according to the length of time to execute the higher-priority OB.

OB call without phase offset



If a phase offset is configured for the lower priority OB and the phase offset is greater than the current execution time of the respective higher priority OB, then the block will be called in a fixed time base.

OB call with phase offset



Parameters

The following table shows the parameters of the "SET_CINT" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_CYCLIC	I, Q, M, D, L or constant	OB number (<32768)
CYCLE	Input	UDINT	I, Q, M, D, L or constant	Time interval in microseconds
PHASE	Input	UDINT	I, Q, M, D, L or constant	Phase offset
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Error code (W#16#....)	Description
0	No error
8090	OB does not exist or is of the wrong type
8091	Incorrect time interval
8092	Incorrect phase offset
80B2	No event assigned to OB

QRY_CINT: Query cyclic interrupt parameters

Description

You can use this instruction to query the current parameters of a cyclic interrupt OB. The cyclic interrupt OB is identified using the OB_NR parameter.

The values of the queried cyclic interrupt parameters correspond to those at the time the "QRY_CINT" instruction is executed.

Parameters

The following table shows the parameters of the "QRY_CINT" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_CYCLIC (INT)	I, Q, M, D, L or constant	OB number (<32768) or symbolic addressing via the name of the OB (e.g. OB_MyOB)
CYCLE	Output	UDINT	I, Q, M, D, L	Time interval in microseconds
PHASE	Output	UDINT	I, Q, M, D, L	Phase offset
STATUS	Output	WORD	I, Q, M, D, L	Status of the cyclic interrupt: • Bit 0 to bit 4: see parameter STATUS • Other bits: Always "0"
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter STATUS

Bit	Value	Meaning
0	0	The CPU is in RUN mode.
	1	The CPU is in startup.
1	0	The cyclic interrupt is enabled.
	1	The cyclic interrupt is delayed.
2	0	The cyclic interrupt is not enabled or has expired.
	1	The cyclic interrupt is enabled.
3	0	-
	1	-
4	0	An OB with the specified number does not exist.
	1	An OB with the specified number exists.
Other bits		Always "0"

Parameter RET_VAL

If an error occurs, the relevant error code will be displayed in the RET_VAL parameter and the STATUS parameter is set to "0".

Error code (W#16#....)	Description
0	No error
8090	OB does not exist or is of the wrong type
80B2	No event assigned to OB

Time-of-day interrupt

SET_TINTL: Set time-of-day interrupt

Description

You can use this instruction to set up the start date and time-of-day of the time-of-day interrupt organization blocks from the user program without making settings in the hardware configuration.

With the SDT parameter, you specify the start date and time-of-day. With the PERIOD parameter, you can specify the cycle at which the instruction call is to be repeated (e.g., daily, once per week). If you set the repetition period to "monthly", you may only specify a day between 1. and 28. for the start date. Days 29 to 31 must not be assigned because a hardware interrupt would not be called in February, for example. If you want to initiate the time-of-day interrupt at the end of each month, use the "End of month" function.

With the ACTIVATE parameter, you specify whether the settings made for the organization block are to be applied directly (ACTIVATE = true) or only after "ACT_TINT (Page 1737)" for the time-of-day interrupt organization block is called (ACTIVATE = false).

Note

When calling time-of-day interrupt organization blocks with a start time within the second hour during changeover from daylight saving time to standard time, also use a time-delay interrupt during the first hour of the time changeover.

Parameters

The following table shows the parameters of the "SET_TINTL" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD (INT)	I, Q, M, D, L or constant	Number of the OB started at time SDT+ multiple of PERIOD (OB 10 to OB 17).
SDT	Input	DTL	D	Start date and time-of-day. The seconds and milliseconds of the start time-of-day are ignored and set to "0".
LOCAL	Input	BOOL	I, Q, M, D, L or constant	• LOCAL = true: Use local time • LOCAL = false: Use system time

Parameter	Declaration	Data type	Memory area	Description
PERIOD	Input	WORD	I, Q, M, D, L or constant	Period from starting point SDT onwards: • W#16#0000 = once • W#16#0201 = once every minute • W#16#0401 = once hourly • W#16#1001 = once daily • W#16#1201 = once weekly • W#16#1401 = once monthly • W#16#1801 = once yearly • W#16#2001 = at month's end
ACTIVATE	Input	BOOL	I, Q, M, D, L or constant	• ACTIVATE = true: Execute instruction • ACTIVATE = false: Execute instruction only when "ACT_TINT (Page 1737)" is called
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, then the actual parameter of RET_VAL contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
8091	Incorrect parameter SDT
8092	Incorrect parameter PERIOD
80A1	The set start time is in the past. (This error code occurs only when PERIOD = W#16#0000.)
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 1222)

CAN_TINT: Cancel time-of-day interrupt

Description

You can use this instruction to delete the start date and time-of-day of a specified time-of-day interrupt organization block. This deactivates the time-of-day interrupt, and the organization block is no longer called.

If you want to use the time-of-day interrupt again, you must first reset the start time ("SET_TINTL (Page 1735)" instruction) and then activate the time-of-day interrupt ("ACT_TINT (Page 1737)" interrupt).

Parameters

The following table shows the parameters of the instruction "CAN_TINT":

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD (INT)	D, L or constant	Number of the OB whose start date and time-of-day are to be deleted.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
80A0	No start date/time specified for the time-of-day interrupt OB
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 1222)

ACT_TINT: Enable time-of-day interrupt

Description

You use the instruction to activate a time-of-day interrupt organization block.

Parameters

The following table shows the parameters of the instruction "ACT_TINT":

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD (INT)	I, Q, M, D, L or constant	Number of the OB to be activated.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
80A0	Start date and time-of day not set for the relevant time-of-day interrupt OB.
80A1	The activated time lies in the past; error occurs for execution "once".
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 1222)

QRY_TINT: Query status of time-of-day interrupt

Description

You can use this instruction to display the status of a time-of-day interrupt organization block in the STATUS output parameter.

Parameters

The following table shows the parameters of the instruction "QRY_TINT":

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD (INT)	I, Q, M, D, L or constant	Number of the OB that will be queried for status (OB 10 to OB 17).
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, then the actual parameter of RET_VAL contains an error code.
STATUS	Output	WORD	I, Q, M, D, L	Status of the time-of-day interrupt; see following table.

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter STATUS

If an error occurs (see RET_VAL parameter), "0" is output in the STATUS parameter.

Bit	Value	Meaning
0	0	In Run.
	1	During startup.
1	0	The time-of-day interrupt is enabled.
	1	The time-of-day interrupt is disabled.
2	0	Time-of-day interrupt is not activated or has elapsed.
	1	The time-of-day interrupt is activated.
4	0	An OB with an OB number as specified at OB_NR parameter does not exist.
	1	An OB with an OB number as specified at OB_NR parameter does exist.
6	0	Base for the time-of-day interrupt is the basic time
	1	Base for the time-of-day interrupt is the local time
Other		Always "0"

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 1222)

Time-delay interrupt

Using time-delay interrupts

Definition

After you have called the "SRT_DINT (Page 1741)" instruction, the operating system generates an interrupt after the specified delay time has elapsed, in other words, the assigned time-delay interrupt OB is called.

Prerequisites for the call

Before a time-delay interrupt can be called by the operating system, the following conditions must be met:

- The time-delay interrupt OB must be started by the "SRT_DINT (Page 1741)" instruction.
- The time-delay interrupt OB must not be deselected during configuration.
- The time-delay interrupt OB must exist in the CPU.

Purpose of the instructions "SRT_DINT", "CAN_DINT" and "QRY_DINT"

You use the instructions to

- Start time-delay interrupts ("SRT_DINT (Page 1741)")
- Cancel time-delay interrupts ("CAN_DINT (Page 1742)")
- Query time-delay interrupts ("QRY_DINT (Page 1743)")

Effects on the time-delay interrupt

The following table lists a number of different situations and explains the effect they have on a time-delay interrupt.

If ...	and ...	Then ...
A time-delay interrupt is started (by calling "SRT_DINT (Page 1741)")	The time-delay interrupt has already started,	The delay time is overwritten; the time-delay interrupt is started again.
	The time-delay interrupt OB does not exist at the time of the call,	The operating system generates a priority class error (calls OB 85). If OB 85 does not exist, the CPU changes to STOP.
	The interrupt is started in a startup OB and the delay time elapses before the CPU changes to RUN,	The call of the time-delay interrupt OB is delayed until the CPU is in RUN mode.
The delay time has elapsed,	A previously started time-delay interrupt OB is still being executed,	The operating system generates a time error (calls OB 80). If OB 80 does not exist, the CPU changes to STOP.

Response to warm restart and cold restart

During a warm restart or a cold restart, all the time-delay interrupt settings made in the user program by means of instructions are cleared.

Starting in a startup OB

A time-delay interrupt can be started in a startup OB. Two conditions must be satisfied to call the time-delay OB:

- The delay time must have elapsed.
- The CPU must be in the RUN mode.

If the delay time has elapsed and the CPU is not yet in the RUN mode, the time-delay interrupt OB call is delayed until the CPU is in RUN mode. The time-delay interrupt OB is then called before the first instruction in OB Main [OB 1] is executed.

SRT_DINT: Start time-delay interrupt

Description

You use this instruction to start a time-delay interrupt that calls a time-delay interrupt OB after the delay time specified in the DTIME parameter has elapsed. The delay time is started when a negative edge is generated in the EN enable input.

If the countdown of the delay time is interrupted, the organization block specified in the OB_NR parameter is not executed.

Accuracy

The maximum time between the "SRT_DINT" instruction call and the start of the time-delay interrupt OB is one millisecond less than the assigned delay time, provided that no interruption events delay the call.

Parameters

The following table shows the parameters of the "SRT_DINT" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_DELAY (INT)	D, L or constant	Number of the OB to be executed after a delay time
DTIME	Input	TIME	I, Q, M, D, L or constant	Delay time (1 to 60000 ms) You can realize longer times, for example, by using a counter in a time-delay interrupt OB.
SIGN	Input	WORD	I, Q, M, D, L or constant	Note: You must assign a value to this parameter upon call. However, the value has no significance.
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error
8090	Incorrect OB_NR parameter
8091	Incorrect DTIME parameter
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 1222)

CAN_DINT: Cancel time-delay interrupt

Description

You use this instruction to cancel a started time-delay interrupt and, thus, also cancel the call of the time-delay interrupt OB that is to be executed after the assigned delay time. You specify the number of the organization block whose call is to be canceled in the OB_NR parameter.

Parameters

The following table shows the parameters of the "CAN_DINT" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_DELAY (INT)	I, Q, M, D, L or constant	Number of the OB whose call will be canceled
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error
8090	Incorrect OB_NR parameter

QRY_DINT: Query time-delay interrupt status

Description

You can use this instruction to query the status of a time-delay interrupt.

Parameters

The following table shows the parameters of the instruction "QRY_DINT":

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_DELAY (INT)	D, L or constant	Number of the OB whose status is to be queried.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code. The value "0" is displayed in the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status of the time-delay interrupt, see following table.

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter STATUS

Bit	Value	Meaning
0	0	Operating system in RUN
	1	Operating system in startup
1	0	Time-delay interrupt is enabled by the operating system.
	1	Time-delay interrupt is disabled.
2	0	Time-delay interrupt is not activated or has elapsed.
	1	Time-delay interrupt is activated.
3	-	-
4	0	Time-delay interrupt OB with the specified number does not exist.
	1	Time-delay interrupt OB with the specified number exists.
Other bits		Always "0"

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect information in the OB_NR parameter
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 1222)

Asynchronous error event

DIS_AIRT: Delay execution of higher priority interrupts and asynchronous error events

Description

You use "DIS_AIRT" to delay the processing of interrupt OBs whose priority are higher than the priority of the current organization block.

You can call "DIS_AIRT" multiple times in an organization block. The "DIS_AIRT" calls are counted by the operating system. Processing is delayed more and more each time "DIS_AIRT" is executed. To cancel a delay, you must execute the "EN_AIRT (Page 1745)" instruction. To cancel all delays, the number of "EN_AIRT (Page 1745)" executions must be equal to the number of "DIS_AIRT" calls.

You can query the number of delays in the RET_VAL parameter of the "DIS_AIRT" instruction. If the value in the RET_VAL parameter is "0", there are no delays.

Parameters

The following table shows the parameters of the "DIS_AIRT" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Number of delays

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

EN_AIRT: Enable execution of higher priority interrupts and asynchronous error events

Description

You use "EN_AIRT" to enable processing of organization blocks when interrupts occur that have been delayed by the "DIS_AIRT (Page 1744)" instruction.

When "EN_AIRT" is executed, you cancel a processing delay that was registered by the operating system when "DIS_AIRT (Page 1744)" was called. To cancel all delays, the number of "EN_AIRT" executions must be equal to the number of "DIS_AIRT (Page 1744)" calls. If, for example, you have called "DIS_AIRT (Page 1744)" five times and thereby also delayed the processing five times, you must call the "EN_AIRT" instruction five times in order to cancel all five delays.

You can query the number of interrupt delays that have not yet been enabled after the execution of "EN_AIRT" in the RET_VAL parameter of the "EN_AIRT" instruction. The value "0" in the RET_VAL parameter means that all delays enabled by "DIS_AIRT (Page 1744)" have been cancelled.

Parameters

The following table shows the parameters of the "EN_AIRT" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Number of configured delays

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

9.8.3.5 Diagnostics

LED: Read LED status

Description

You can use the "LED" instruction to read out the status (e.g., "On" or "Off") of a particular module LED.

- With the LADDR parameter, you address the CPU or the interface.
- With the LED parameter, you select the module LED whose current status is to be read out using the instruction.
- The RET_VAL parameter outputs the status of the selected LED when the instruction is called. Depending on the LED selected, only certain status information may be displayed. For example, some LEDs have only color information. Refer to the hardware documentation of the respective module for the available status information of a particular LED.

Parameters

The following table shows the parameters of the "LED" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, L or constant	Identification number of the CPU or interface. The number is automatically assigned and stored in the CPU properties or the interface in the hardware configuration.
LED	Input	UINT	I, Q, M, D, L or constant	Identification number of the LED: • 1: STOP/RUN • 2: ERROR • 3: MAINT (maintenance) • 4: Redundant • 5: Link (green) • 6: Rx/Tx (yellow)
RET_VAL	Return	INT	I, Q, M, D, L	Status of LED

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

RET_VAL	Description
0 to 9	LED status: • 0 = LED does not exist • 1 = Permanently switched off • 2 = Color 1 (e.g., for LED STOP/RUN: green) permanently ON • 3 = Color 2 (e.g., for LED STOP/RUN: orange) permanently ON • 4 = Color 1 flashing at 2 Hz • 5 = Color 2 flashing at 2 Hz • 6 = Colors 1 and 2 flashing alternately at 2 Hz • 7 = LED is active, color 1 • 8 = LED is active, color 2 • 9 = LED exists, but status information not available
8091	Module addressed with the LADDR parameter does not exist.
8092	A module that does not support LEDs was addressed with the LADDR parameter.
8093	The identification number specified in the LED parameter is not defined.
80Bx	The CPU specified in the LADDR parameter does not support the "LED" instruction.

DeviceStates: Read module status information of an IO system

Description

You use the "DeviceStates" instruction to output the status of the modules of an IO system. The status information is selected with the LADDR and MODE parameters:

- With the LADDR parameter, you select the IO system.
- With the MODE parameter, you select which status information is to be output.

The module status read out with the "DeviceStates" instruction is also displayed in the Diagnostics view of the modules.

Parameters

The following table shows the parameters of the "DeviceStates" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_ IOSYSTEM	I, Q, M, L or constant	Identification number of the IO system
MODE	Input	UINT	I, Q, M, D, L or constant	Selection of status information to be read
RET_VAL	Return	INT	I, Q, M, D, L	Status of instruction (see table parameter MODE)
STATE	InOut	VARIANT	I, Q, M, D, L	Buffer for the IO system status; The pointer can refer to the following data types: BOOL, BYTE, WORD, DWORD, LWORD or an Array of [...] of these data types.

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter MODE

With the MODE parameter, you select which status information is to be output based on the following numbers.

MODE	Description
1	Configuration of module/device is active or not yet complete
2	Module defective
3	Module disabled
4	Module exists
5	There is a problem in the module.

Parameter STATE

With the STATE parameter, the status of the modules selected with the MODE parameter is output. The status information is output as a bit character string. The length of the bit character string depends on the I/O system:

- For PROFIBUS-DP the length of the character string is 128 bits.
- For PROFINET-IO the length of the character string is 1024 bits.

If the status selected using MODE applies to a module, the corresponding bit of the module is set to "1". For example, if an error has occurred in the third module, the third bit is set to "1". The "0" bit of the bit character string summarizes the status information for all modules of an I/O system:

- Bit 0 = 0: No errors have occurred in any module/all bits of the bit character string are set to "0".
- Bit 0 = 1: An error has occurred in at least one module/at least one bit of the bit character string was set to "1".

Parameter RET_VAL

Error code (W#16#...)	Description
0	No error
8091	LADDR does not exist
8092	LADDR does not address an IO system
80Bx	The module specified in the "LADDR " parameter does not support the "DeviceStates" instruction.
8452	The complete status information does not fit in the tag configured in the STATE parameter. The result is only output up to the byte length of the tag used.

ModuleStates: Read module status information of a module

Description

You use the "ModuleStates" instruction to read out the status information of a module. You select the status information with the LADDR and MODE parameters:

- Use the LADDR parameter to select the module.
- Use the MODE parameter to select which information is to be output.
- The STATE parameter indicates the status of the module selected with the MODE parameter.

Parameters

The following table shows the parameters of the "ModuleStates" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, L or constant	Identification number of the module
MODE	Input	UINT	I, Q, M, D, L or constant	Selection of status information to be read
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction
STATE	InOut	VARIANT	I, Q, M, D, L	Buffer for the status of the module

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter MODE

With the MODE parameter, you select which status information is to be output based on the following numbers.

MODE	Description
1	Configuration of module is active or not yet complete
2	Module defective
3	Module disabled
4	Module exists
5	There is a problem in the module.

Parameter STATE

Any bit string (BOOL, BYTE, WORD) or an Array of a bit string can be used as a data type (e.g. Array of BYTE). The length of the bit string depends on the device you use. The maximum length is 128 bit. If the status selected using MODE applies to a module, the corresponding bit of the module sub-module is set to "1". For example, if an error has occurred for the sub-module in the third slot of the module, the third bit is set to "1". The "0" bit of the bit character string summarizes the status information for all devices:

- Bit 0 = 0: An error did not occur for any module. All the bits of the bit character string are set to "0".
- Bit 0 = 1: An error occurred for at least one sub-module of the module. At least one bit of the bit character string was set to "1".

Parameter RET_VAL

Error code (W#16#...)	Description
0	No error
8091	LADDR does not exist
8092	LADDR does not address an IO module
80Bx	The module specified in the "LADDR" parameter does not support the "ModuleStates" instruction.
8452	The complete status information does not fit in the tag configured in the STATE parameter. The result is only output up to the bit length of the tag used.

GET_DIAG: Read diagnostic information

Description

You can use the "GET_DIAG" instruction to read out the diagnostic information of a hardware object. The hardware object is selected with the LADDR and CHANNEL parameters. With the MODE parameter, you select which diagnostic information is to be read out.

Parameters

The following table shows the parameters of the "GET_DIAG" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	UINT	I, Q, M, D, L or constant	Use the MODE parameter to select which diagnostic data is to be output.
LADDR	Input	HW_ANY (WORD)	I, Q, M, L or constant	Hardware ID of the device.
CHANNEL	Input	UINT	I, Q, M, D, L or constant	Channel number
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction
CNT_DIAG	Output	UINT	I, Q, M, D, L	Number of output diagnostic details
DIAG	InOut	VARIANT	I, Q, M, D, L	Pointer to data area for storage of diagnostic information of the selected mode.
DETAILS	InOut	VARIANT	I, Q, M, D, L	Pointer to data area for storage of diagnostic details in accordance with the selected mode.

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter MODE

Depending on the value at the MODE parameter, different diagnostics data is output at the DIAG, CNT_DIAG and DETAILS output parameters.

MODE	Description	DIAG	CNT_DIAG	DETAILS
0	Output of all supported diagnostic information for a module as DWORD, where Bit X=1 indicates that mode X is supported.	Bit string of the supported modes as DWORD, where Bit X=1 indicates that mode X is supported.	0	-
1	Output of the inherent status of the addressed hardware object.	Diagnostics status. Output of the inherent status of the addressed hardware object in accordance with the DIS structure.	0	-
2	Output of the status of all subordinate modules of the addressed hardware object.	Diagnostics status. Output in accordance with the DIS structure.	1	Module status information in accordance with the DiagnosticsDetails structure.
3	Output of the I/O status of the addressed hardware object.	Diagnostics status. Output in accordance with the DIS structure.	Number of diagnostics data details that were output.	Status of the channels (DiagnosticsDetails).
4	Output of the I/O status of all subordinate modules of the addressed hardware object.	Output of diagnostics data in accordance with the DNN structure	0	-

DIS structure

With parameter MODE = 1 to 3, the diagnostics information is output in accordance with the DIS structure. The following table shows the meaning of the individual parameter values.

Parameter	Data type	Value	Description
OwnState	UINT16	Enum	The value of the parameter Ownstate describes the maintenance status of the module.
		0	No fault
		1	The module or device is disabled.
		2	Maintenance required
		3	Maintenance demanded
		4	Error
		5	The module or the device cannot be reached from the CPU (valid for modules and devices below a CPU).
		6	Inputs/outputs are not available.
		7	-
MaintenanceState	DWORD	Enum	
		0	No maintenance required

Parameter	Data type	Value	Description
		1	The module or device is disabled.
		2	-
		3	-
		4	-
		5	Maintenance required
		6	Maintenance demanded
		7	Error
		8	Status unknown / error in subordinate module
		9	-
		10	Inputs/outputs are not available.
IO State	UINT16	Bit array	I/O status of the module
		0	Bit 0 = 1: No maintenance required
		1	Bit 1 = 1: The module or device is disabled.
		2	Bit 2 = 1: Maintenance required
		3	Bit 3 = 1: Maintenance demanded
		4	Bit 4 = 1: Error
		5	Bit 5 = 1: The module or the device cannot be reached from the CPU (valid for modules and devices below a CPU).
		6	Inputs/outputs are not available.
		7	Qualifier; bit 7 = 1, if bit 0, 2, or 3 are set
		8 to 15	Reserved (always = 0)
Componentstate Detail	DWORD	Bit array	Status of the module sub-modules: • Bit 0 to 15: Status message of the module • Bit 16 to 31: Status message of the CPU
		0 to 2 (enum)	Additional information: • Bit 0: No additional information • Bit 1: Transfer not permitted
		3	Bit 3 = 1: At least one channel supports qualifiers for diagnostics
		4	Bit 4 = 1: Maintenance required for at least one channel or one component.
		5	Bit 5 = 1: Maintenance demanded for at least one channel or one component.
		6	Bit 6 = 1: Error in at least one channel or one component.
		7 to 10	-
		11 to 14	Bit 11 = 1: PNIO - sub-module correct Bit 12 = 1: PNIO - replacement module Bit 13 = 1: PNIO - incorrect module Bit 14 = 1: PNIO - module disconnected
		15	-

Parameter	Data type	Value	Description
		16 to 31	Status information for modules generated by the CPU: Bit 16 = 1: Module disabled Bit 17 = 1: CiR operation active Bit 18 = 1: Input not available Bit 19 = 1: Output not available Bit 20 = 1: Overflow diagnostics buffer Bit 21 = 1: Diagnostics not available Bit 22 - 31: Reserved (always 0)
OperatingState	UInt16	Enum	
		0	-
		1	In STOP / Firmware update
		2	In STOP / reset memory
		3	In STOP / self start
		4	In STOP
		5	Memory reset
		6	In START
		7	In RUN
		8	-
		9	In HOLD
		10	-
		11	-
		12	Module defective
		13	-
		14	No power
		15	CiR
		16	In STOP / without ODIS
		17	In
		18	
		19	
		20	

DiagnosticsDetail structure

With parameter MODE = 2 or 3, the diagnostics information details are output in accordance with the DiagnosticsDetail structure. The following table shows the meaning of the individual parameter values.

Parameter	Data type	Description
ChannelNumber	UInt	Channel number
Properties	Word	
ALID	UInt	Identification ID of alarm
Qualifier	DWord	Qualifier of diagnostic data

Parameter	Data type	Description
ErrorType	UDInt	Channel error type
ExtErrorType	UDInt	Extended channel error type
AddValue_1	UInt	Additional value
AddValue_2	UInt	Additional value
AddValue_3	UInt	Additional value
AddValue_4	UInt	Additional value

DNN structure

With parameter MODE = 4, the diagnostics information details are output in accordance with the DNN structure. The following table shows the meaning of the individual parameter values.

Parameter	Data type	Value	Description
SubordinateState	UINT	Enum	Status of the subordinate module (see parameter OwnState of the DIS structure)
SubordinateIOState	WORD	Bitarray	Status of the inputs and outputs of the subordinate module (see parameter IO State of the DIS structure)
DNNmode	WORD	Bitarray	• Bit 0 = 0: Diagnostics enabled • Bit 0 = 1: Diagnostics disabled • Bit 1 to 15: Reserved

Parameter RET_VAL

Error code (W#16#...)	Description
0	No error
n	The data area in the DETAILS parameter is too small. Not all details of the diagnostic data can be output.
8080	Value in the MODE parameter is not supported.
8081	Type in the DIAG parameter is not supported with the selected mode (parameter MODE).
8082	Type in the DETAILS parameter is not supported with the selected mode (parameter MODE).
8090	LADDR does not exist
8091	The selected channel in the CHANNEL parameter does not exist.
80C1	Insufficient resources for parallel execution.

9.8.3.6 Pulse

CTRL_PWM: Pulse-width modulation

Description

You can use the "CTRL_PWM" instruction to enable and disable a pulse output supported by the CPU using the software.

Note

Pulse output parameters are assigned exclusively in the device configuration and not using the "CTRL_PWM" instruction. Any change of parameters that is intended to have an effect on the CPU must therefore be made while the CPU is in STOP mode.

You enter the hardware ID of the pulse output you want to control with the instruction in the PWM input. Error-free execution of the instruction is possible only when the specified pulse output is enabled in the hardware configuration.

Only tags of "HW_PWM" data type can be specified in the PWM input. The hardware data type HW_PWM has a length of one WORD.

The pulse output is enabled when the bit in the ENABLE input of the instruction is set. If ENABLE has the value TRUE, the pulse output generates pulses that have the properties defined in the device configuration. When the bit in the ENABLE input is reset or the CPU changes to STOP, the pulse output is disabled and no more pulses are generated.

The "CTRL_PWM " instruction is only executed if the signal state in the EN input is "1".

Since the S7-1200 enables the pulse output when the "CTRL_PWM" instruction is executed, BUSY at S7-1200 always has the value FALSE.

The ENO enable output is set only when the EN enable input has signal state "1" and no errors have occurred during execution of the instruction.

Note

Use of the force table for PWM and PTO

Digital inputs and outputs that are used for PWM and PTO cannot be forced. Digital inputs and outputs that were assigned via device configuration cannot be controlled by either the force table or the monitoring table.

Parameters

The following table shows the parameters of the "CTRL_PWM" instruction:

Parameter	Declaration	Data type	Memory area	Description
PWM	Input	HW_PWM	I, Q, M, L or constant	Hardware ID of the pulse generator
ENABLE	Input	BOOL	I, Q, M, D, L or constant	The pulse output is enabled when ENABLE = TRUE and disabled when ENABLE = FALSE.

Parameter	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	Processing status
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter STATUS

Error code (W#16#...)	Description
0	No error
80A1	Hardware ID of the pulse generator is invalid

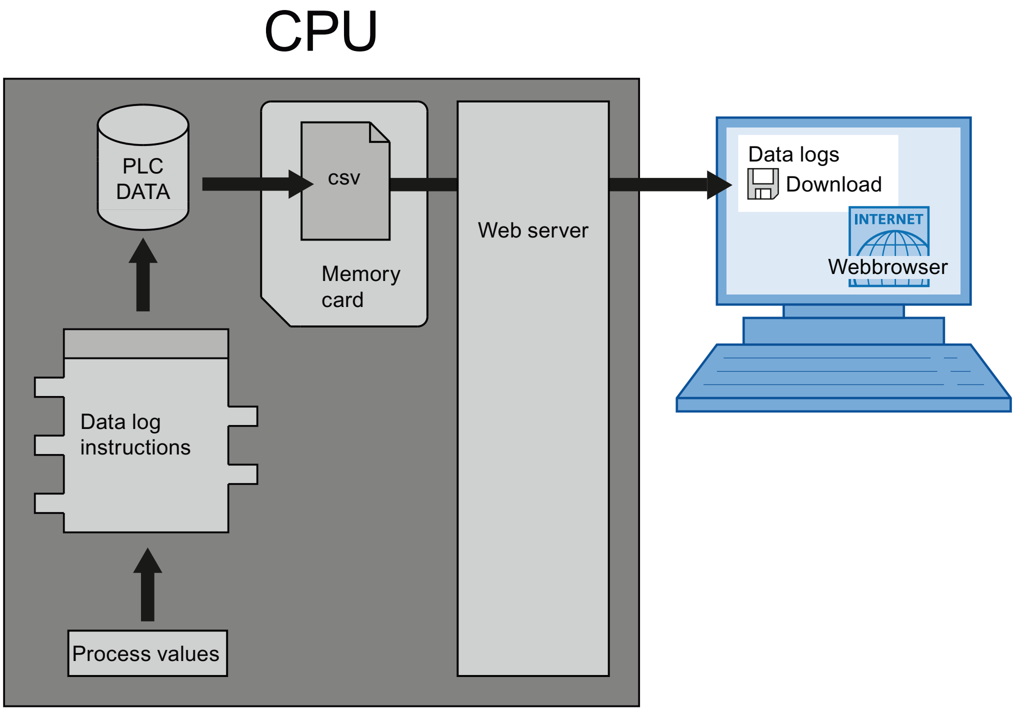
9.8.3.7 Data logging

Data logging - Overview

Saving process values

Data logging instructions are used in the user program to save process values to data logs. Data logs are saved in the load memory in standard CSV (comma separated value) format.

The data logging instructions are used in your program to create and open a data log, to write an entry, and to close the data log files. You decide what program values will be placed in the data log when creating the data buffer. The data buffer is used as a temporary memory for new data log entries. New values must be to moved to the buffer via the program during runtime of the user program. If the values are updated, the "DataLogWrite" instruction can be executed. In so doing, the data are written from the buffer to a data log data record.



Data log files can be copied to the PC as follows:

- If the PROFINET interface is connected to the PC, you use a Web browser to access the data logs via the Web server. The CPU can be in RUN or STOP mode for this. If the CPU is in "RUN" mode, the program continues running while the Web server is transferring data.
- If there is a memory card in the S7-1200 CPU, you can remove this card and insert it into a standard slot for SD (Secure Digital) cards or MMC (MulitMediaCard) cards on a PC. Use File Manager to transfer the data log files from the memory card to the PC. The CPU goes to "STOP" when you remove the memory card.

Properties of the data log

The data logs are circular logs. New data records are added until the maximum number of data records is reached (RECORD parameter). The next data record then overwrites the "oldest" data record of the data log.

If you want to prevent data records from being overwritten, use the "DataLogNewFile" instruction to create a new data log file based on the current data log. New data records are then written to the new data log file.

Creating a data log

You use the "DataLogCreate" instruction to create a new data log file in the "\Logs" directory in the load memory. Central block parameters are NAME and DATA. The data logs have a name and an ID. The name is the name for the data log and is used as a file name for the cs file (file is stored in the "Logs" folder). The DATA block parameter defines the temporary data buffer for the new data log object. The "DataLogCreate" instruction returns an ID. This ID is used by the other data logging instructions as a reference for the created data log.

The columns and data types of a data record in the data log are generated by the elements of a structure declaration of a PLC data type declaration or an array declaration. Each element of a structure or an array corresponds to one element in a line in the log file.

You can use the HEADER block parameter to assign a header to each column.

Writing to data logs

As a prerequisite to writing a data record to a data log, a data log must be open ("DataLogOpen" instruction). The "DataLogWrite" instruction writes a data record to the data log.

DataLogCreate: Create data log

Description

You use the "DataLogCreate" instruction to create a data log. The data log is saved to the memory card in the "\Archives" folder. You can use the data logging instructions to save process data to the memory card. The amount of data that can be stored in a data log is dependent on the memory space available on the memory card of the CPU used.

You specify the maximum number of data records that can be stored in a data log in the RECORDS parameter. If the specified maximum number of data records in the data log is reached, the oldest data record is overwritten in each case. To prevent overwriting of existing data records, use the "DataLogNewFile (Page 1765)" instruction. The instruction creates a new data log with the same structure when the number of records specified in the RECORDS parameter is reached. The data records are then saved in the new data log.

You can specify the name for the data log in the NAME parameter. The data log is created in the CSV Comma Separated Values format. With the HEADER parameter, you can create an (optional) header for the data log. A comma or a period can be used as a separator.

Once the data log is created, it is opened automatically.

Parameters

The following table shows the parameters of the "DataLogCreate" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D or constant	Execution of instruction on a rising edge.
RECORDS	Input	UDInt	I, Q, M, L, D or constant	Number of data records in the data log
FORMAT	Input	UInt	I, Q, M, L, D or constant	Data format: • 0: Internal (not supported) • 1: CSV (Comma separated values)
TIMESTAMP	Input	UInt	I, Q, M, L, D or constant	Time stamp: • 0: No time stamp • 1: Date and time With the time stamp, an additional header is not required for the data log.
DONE	Output	BOOL	I, Q, M, L, D	Instruction is being executed
BUSY	Output	BOOL	I, Q, M, L, D	Creation of the data log is not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	Error
STATUS	Output	WORD	I, Q, M, L, D	Status parameter (The parameter is set for the duration of one call only. To display the status, you should copy the STATUS parameter to a free data area.) STATUS has the following meaning, depending on the ERROR bit: • ERROR= "0": STATUS has the value W#16#0000: Neither warning nor error • ERROR= "1": An error has occurred, STATUS supplies detailed information on the type of error.
NAME	InOut	String	I, Q, M, L, D or constant	Name of the data log. The specified name is also used as a file name for the csv file. The restrictions for Windows file names apply when assigning the name. The following characters must not be used: "\", "/", ":", "*", "?", "<", ">", "|", "space"
ID	InOut	DataLog	I, Q, M, L, D	Object ID of the data log (output only). The ID of the data log is required for the additional data logging instructions.
HEADER	InOut	VARIANT	I, Q, M, L, D	Header of the CSV file
DATA	InOut	VARIANT	I, Q, M, L, D	Data length of the data log data record

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter HEADER

The HEADER parameter is a VARIANT pointer to a data block that defines a header for the CSV file (header). The header is always the first line in the CSV file representation. When a header is created, the individual columns must each be separated by a comma. ASTRING, Array of BYTE, or Array of CHAR data type can be used for the individual column names. When the Array [...] of type data type is used, a longer character string is possible than when the STRING data type is used. When the STRING data type is used, the length is limited to 254 bytes.

If no header is to be created, do not specify a value in the HEADER parameter.

Parameter DATA

The DATA parameter is a VARIANT pointer to a data block in which the number of columns and their data type are specified for a log data record.

Note the following when creating the data block:

- The number of columns must equal the number of columns defined in the HEADER parameter.
- When STRUCT data type is used, nesting must not be used.
- The tags of the data block can be set as retentive or non-retentive tags. However, the retentive setting must be the same for all tags of the data block.

Parameter STATUS

Error code (W#16#...)	Description
0	No error
7000	No job processing active.
7001	Start of the job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	All internal instance memory is in use.
8090	Invalid file name
8093	Data log already exists
8097	File length exceeds the file system limit
80B3	Load memory not sufficient
80B4	The memory card is read-only
80C1	Too many files open
8453	Invalid format selection
8553	Invalid time stamp
8B51	Invalid data type in the HEADER parameter.
8C51	Invalid data type in the DATA parameter.

DataLogOpen: Open data log

Description

You use the "DataLogOpen" instruction to open an existing data log on the memory card. A data log must be open in order to write new data records to the data log.

Data log opens automatically when you execute the instructions "DataLogCreate (Page 1758)" and "DataLogNewFile (Page 1765)".

Up to 10 data logs can be open simultaneously. You can select the data log to be opened using the ID or name of the data log.

- If you specify the ID and the name of the data log in the ID and NAME parameters, respectively, the data log will be identified based on the ID. The data log name comparison is not carried out.
- If you select the data log using the NAME parameter and no ID is specified, the ID will be displayed in the ID parameter when the data log is opened.
- If you select the data log using the ID parameter and no name is specified, the name will be displayed in the NAME parameter when the data log is opened.

With the MODE parameter, you select whether the data records of the data log are deleted on opening.

Parameters

The following table shows the parameters of the "DataLogOpen" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D or constant	Execution of instruction on a rising edge.
MODE	Input	UInt	I, Q, M, L, D or constant	Mode for opening the data log: • MODE= "0" Retain data records of the data log • MODE= "1" Data records of the data log are deleted, but the header is retained
DONE	Output	BOOL	I, Q, M, L, D	Instruction is being executed
BUSY	Output	BOOL	I, Q, M, L, D	• 0: Instruction executed. • 1: Execution of instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	Error

Parameter	Declaration	Data type	Memory area	Description
STATUS	Output	WORD	I, Q, M, L, D	Status parameter (This parameter is only set for the duration of one call. To display the status, you should copy the STATUS parameter to a free data area.) STATUS has the following meaning, depending on the ERROR bit: • ERROR= "0": STATUS has the value W#16#0000: Neither warning nor error • ERROR= "1": An error has occurred, STATUS supplies detailed information on the type of error.
NAME	InOut	String	I, Q, M, L, D or constant	(File) name of the data log.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter STATUS

Error code (W#16#...)	Description
0	No error
2	Data log file was already opened by this application
7000	No job processing active.
7001	Start of the job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	All internal instance memory is in use.
8090	Data log definition and data log control object are inconsistent.
8091	Data log definition and data log data are inconsistent.
8092	Data log does not exist
80B0	Data log file not open.
80C0	Data log file is locked

DataLogWrite: Write data log

Description

You use the "DataLogWrite" instruction to create a data log data record. With the ID parameter, you select the data log in which the data record is to be written. To create a new data record, the data log must be open. The instruction creates a new data record in the format that was specified in the DATA parameter when the data log was created. You must transfer the data values to be transferred to the data log to the data block in the DATA parameter during runtime. When the "DataLogWrite" instruction is executed, these values are copied to the data log.

CAUTION
Data log data loss when the power supply to the CPU is interrupted
If the power supply is interrupted during execution of the "DataLogWrite" instruction, the data record to be transferred is lost.

Parameters

The following table shows the parameters of the "DataLogWrite" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D or constant	Execute function on a rising edge.
DONE	Output	BOOL	I, Q, M, L, D	Instruction is being executed
BUSY	Output	BOOL	I, Q, M, L, D	Creation of the data log is not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	Error
STATUS	Output	WORD	I, Q, M, L, D	Status parameter (The parameter is set for the duration of one call only. To display the status, you should copy the STATUS parameter to a free data area.) STATUS has the following meaning, depending on the ERROR bit: • ERROR= "0": STATUS has the value W#16#0000: Neither warning nor error • ERROR= "1": An error has occurred, STATUS supplies detailed information on the type of error.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter STATUS

Error code (W#16#...)	Description
0	No error
0001	Last possible data record created at the end of the file. Creation of another data record will overwrite an older data record.
7000	No job processing active.
7001	Start of the job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	All internal instance memory is in use.
8080	Data log definition and data log control object are inconsistent.
8090	Data log definition and data log data are inconsistent.
8092	Data log does not exist.
80B0	Data log not open.
80C0	Data log file is locked.

DataLogClose: Close data log

Description

You use the "DataLogClose" instruction to close an open data log. You select the data log using the ID parameter.

Note

Closing data logs automatically

The data log is closed automatically when the CPU goes to STOP or if there is a restart.

Parameters

The following table shows the parameters of the "DataLogClose" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D or constant	Execute function on a rising edge.
DONE	Output	BOOL	I, Q, M, L, D	Instruction is being executed
BUSY	Output	BOOL	I, Q, M, L, D	Closing of the data log is not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	Error

Parameter	Declaration	Data type	Memory area	Description
STATUS	Output	WORD	I, Q, M, L, D	Status parameter (This parameter is only set for the duration of one call. To display the status, you should copy the STATUS parameter to a free data area.) STATUS has the following meaning, depending on the ERROR bit: • ERROR= "0": STATUS has the value W#16#0000: Neither warning nor error • ERROR= "1": An error has occurred, STATUS supplies detailed information on the type of error.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter STATUS

Error code (W#16#...)	Description
0	No error
1	Data log not open
7000	No job processing active.
7001	Start of the job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8092	Data log does not exist

DataLogNewFile: Data log in new file

Description

You use the "DataLogNewFile" instruction to create a new data log with the same properties as an existing data log. This enables the contents of a data log to be received.

When the instruction is called, it creates a new data log in the load memory (memory card) with the name defined in the NAME parameter. With the ID parameter (input), you specify the ID of the old data log whose properties you want to apply to the new data log. With the ID parameter (output), the ID of the new data log is output.

You specify the file size of the new data log with the RECORDS parameter of the instruction.

Parameter

The following table shows the parameters of the "DataLogNewFile" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D or constant	Execute function on a rising edge.
RECORDS	Input	UDInt	I, Q, M, L, D or constant	Number of data records in the data log
NAME	InOut	String	I, Q, M, L, D or constant	File name of the new data log
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed
BUSY	Output	BOOL	I, Q, M, L, D	Creation of the data log is not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	Error
STATUS	Output	WORD	I, Q, M, L, D	Status parameter (This parameter is only set for the duration of one call. To display the status, you should copy the STATUS parameter to a free data area.) STATUS has the following meaning, dependent on the ERROR bit: • ERROR= "0": STATUS has the value W#16#0000: Neither warning nor error • ERROR= "1": An error has occurred, STATUS supplies detailed information on the type of error.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log • Input: ID of the existing data log • Output: ID of the new data log

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter STATUS

Error code (W#16#...)	Description
0	No error
7000	No job processing active.
7001	Start of the job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	All internal instance memory is in use.
8090	Invalid file name
8091	Path does not exist.

Error code (W#16#...)	Description
8092	Source data log does not exist
8093	New data log already exists
8097	File length exceeds the file system limit
80B3	Load memory not sufficient
80B4	The memory card is read-only
80C1	Too many files open

9.8.3.8 Data block functions

READ_DBL: Read from data block in the load memory

Description

With the instruction you copy a DB or an area of a DB in load memory (Micro Memory Card) to the data area of a destination DB. The destination DB must be relevant for execution; that is, it must not be created with the attribute UNLINKED. The content of the load memory is not changed during the copy process.

To ensure data consistency, you must not change the destination area while "READ_DBL" is being executed (i.e., as long as the BUSY parameter has the value TRUE).

The following restrictions apply to the SRCBLK and DSTBLK parameters (source and destination blocks):

- You must be able to divide the length of the VARIANT pointer by eight.
- For a VARIANT pointer of type STRING, the length must be equal to 1.
- The source and destination block must have been created with the same block access, i.e. both must use either the access type "Optimized" or "Standard - compatible with S7-300/400".

Note

""READ_DBL" is processed asynchronously. Therefore, it is not suitable for frequent (or cyclical) reading of tags in the load memory.

Once started, a job is always completed. If the maximum number of simultaneously active "READ_DBL" instructions is reached and you call "READ_DBL" once again at this time in a priority class having higher priority, error code W#16#80C3 will be returned. Consequently it does not make sense to restart the high-priority job right away.

Functional description

The "READ_DBL" instruction works asynchronously, that is, its execution extends over multiple calls. You start the job by calling "READ_DBL" with REQ = 1.

The RET_VAL and BUSY output parameters indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 1220)

Parameter

The following table shows the parameters of the instruction "READ_DBL":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request
SRCBLK	Input	VARIANT	D	Pointer to data block in the load memory that is to be read from
RET_VAL	Return	INT	I, Q, M, D, L	Error information
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.
DSTBLK	Output	VARIANT	D	Pointer to the data block in the work memory that is to be written to

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error
0081	The destination area is larger than the source area. The source area is written completely to the destination area; the remaining bytes of the destination area will not be changed.
7000	First call with REQ=0: No data transfer active; BUSY has the value "0".
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8051	Data block type error.
8081	The source area is larger than the destination area. The destination area is fully written. The remaining bytes of the source area are ignored.
8093	No data block or a data block that is not in the work memory is specified for the DSTBLK parameter.
80B1	No data block is specified for the SRCBLK parameter, or the data block specified there is not a load memory object.
80B4	DB with F-attribute must not be read.
80C3	The maximum number of simultaneously active "READ_DBL" instructions has already been reached.

Error code (W#16#...)	Description
8251	Source DB of incorrect type.
82B1	Source DB not specified or does not exist.
82C0	The source DB is currently being processed by another instruction or a different communication function.
8551	Destination DB of incorrect type.
85B1	Destination DB not specified or does not exist.
85C0	The destination DB is currently being processed by another instruction or a communication function.
8xyy	General error codes See also: Evaluating errors with output parameter RET_VAL (Page 1222)

WRIT_DBL: Write to data block in the load memory

Description

You use the instruction to transfer the contents of a DB or a DB area from the work memory to a DB or a DB area in the load memory (Micro Memory Card). The source DB must be relevant for execution, which means it must not be created with the attribute UNLINKED.

To ensure data consistency, you must not change the source area while "WRIT_DBL" is being executed (i.e., as long as the BUSY parameter has the value TRUE).

The following restrictions apply to the SRCBLK and DSTBLK parameters (source and destination blocks):

- For a VARIANT pointer of type BOOL, the length must be divisible by 8.
- For a VARIANT pointer of type STRING, the length must be equal to 1.
- The source and destination block must have been created with the same block access, i.e. both must use either the access type "Optimized" or "Standard - compatible with S7-300/400".

The "WRIT_DBL" instruction does not change the checksum of the user program if you write a DB that was created using an instruction. However, when a loaded DB is written, the first entry in this DB changes the checksum of the user program.

Note

"WRIT_DBL" is not suitable for frequent (or cyclical) writing of tags in the load memory. This is because the Micro Memory Card technology limits the number of write accesses that can be made to a Micro Memory Card.

Functional description

The "WRIT_DBL" instruction works asynchronously, that is, its execution extends over multiple calls. You start the job by calling "WRIT_DBL" with REQ =1.

The RET_VAL and BUSY output parameters indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 1220).

Parameter

The following table shows the parameters of the instruction "WRIT_DBL":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Write request
SRCBLK	Input	VARIANT	D	Pointer to the DB in the work memory that is to be read from
RET_VAL	Return	INT	I, Q, M, D, L	Error information
BUSY	Output	BOOL	I, Q, M, D, L	BUSY= 1: The writing process is not yet complete.
DSTBLK	Output	VARIANT	D	Pointer to the data block in the load memory that is to be written to

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error
0081	The destination area is greater than the source area. The source area is written completely to the destination area; the remaining bytes of the destination area will not be changed.
7000	First call with REQ=0: No data transfer active; BUSY has the value "0".
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8051	Data block type error.
8081	The source area is larger than the destination area. **The destination area is fully written. The remaining bytes of the source area are ignored.**
8092	Incorrect operating mode: While "WRIT_DBL" was active, the CPU went into STOP mode. This error code is supplied at the next transition to RUN. Call "WRIT_DBL" again.
8093	No data block or a data block that is not in the work memory is specified for the DSTBLK parameter.
80B1	No data block is specified for the SRCBLK parameter or the data block specified there is not a load memory object.

Error code (W#16#...)	Description
80B4	DB with F-attribute must not be read.
80C3	The maximum number of simultaneously active "WRIT_DBL" instructions has already been reached.
8251	Source DB of incorrect type.
82B1	Source DB not specified or does not exist.
82C0	The source DB is currently being processed by another instruction or a different communication function.
8551	Destination DB of incorrect type.
85B1	Destination DB not specified or does not exist.
85C0	The destination DB is currently being processed by another instruction or a communication function.
8xyy	General error codes See also: Evaluating errors with output parameter RET_VAL (Page 1222)

9.8.4 Technology

9.8.4.1 PID Control

PID_Compact

Description of PID_Compact

Description

The PID_Compact instruction provides a PID controller with integrated tuning for automatic and manual mode.

Call

PID_Compact is called in the constant interval of the cycle time of the calling OB (preferably in a cyclic interrupt OB).

Startup

At the startup of the CPU, PID_Compact starts in the operating mode that was last active. To retain PID_ Compact in "Inactive" mode, set sb_RunModeByStartup = FALSE.

Monitoring of the sampling time PID_Compact

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID_Compact instruction measures the time interval between two calls. This is the current sampling time. On every change of the operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. If the current sampling time deviates too much from this mean value, Error = 0800 hex occurs and PID_Compact switches to "Inactive" mode.

PID_Compact 1.1 is set to "Inactive" mode during controller tuning under the following conditions:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value

In automatic mode, PID_Compact 1.1 is set to "Inactive" mode under the following conditions:

- New mean value >= 1.5 x old mean value
- New mean value <= 0.5 x old mean value

During controller tuning and in automatic mode, PID_Compact 1.0 is set to "Inactive" operating mode under the following conditions:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value
- Current sampling time >= 1.5 x current mean value
- Current sampling time <= 0.5 x current mean value

Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during controller tuning and rounded to a multiple of the cycle time. All other functions of PID_Compact are executed at every call.

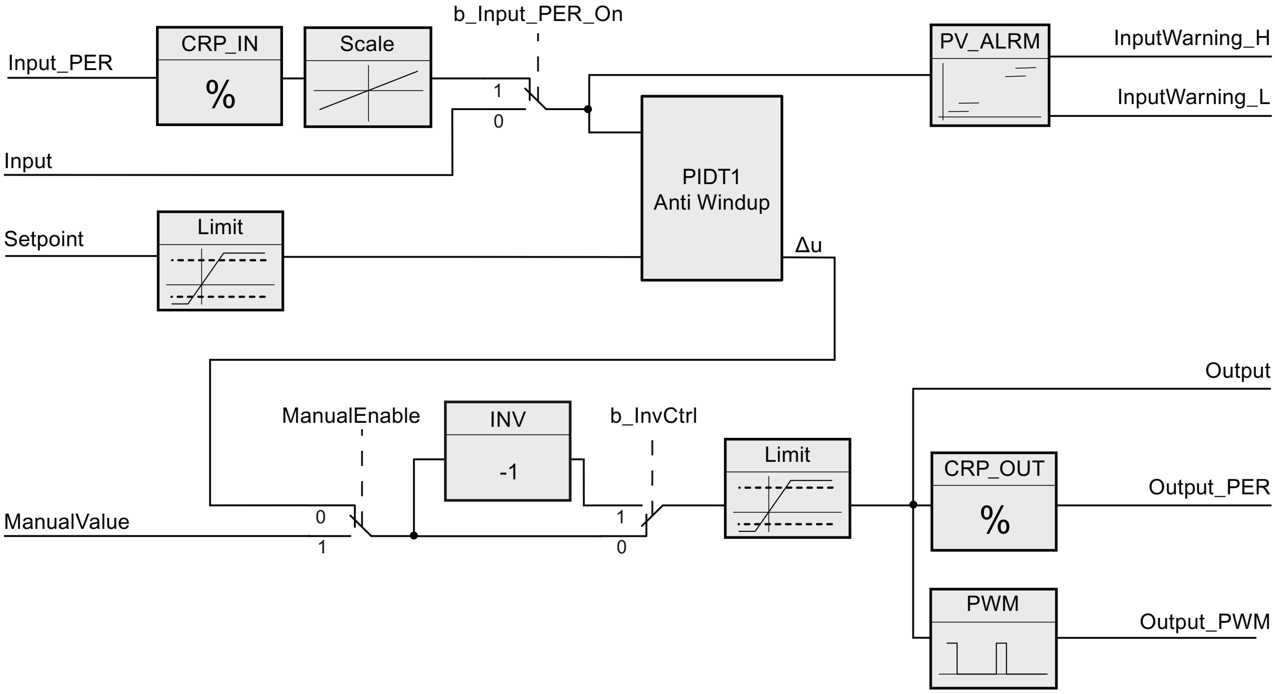
PID algorithm

PID_Compact is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The following equation is used to calculate the output value.

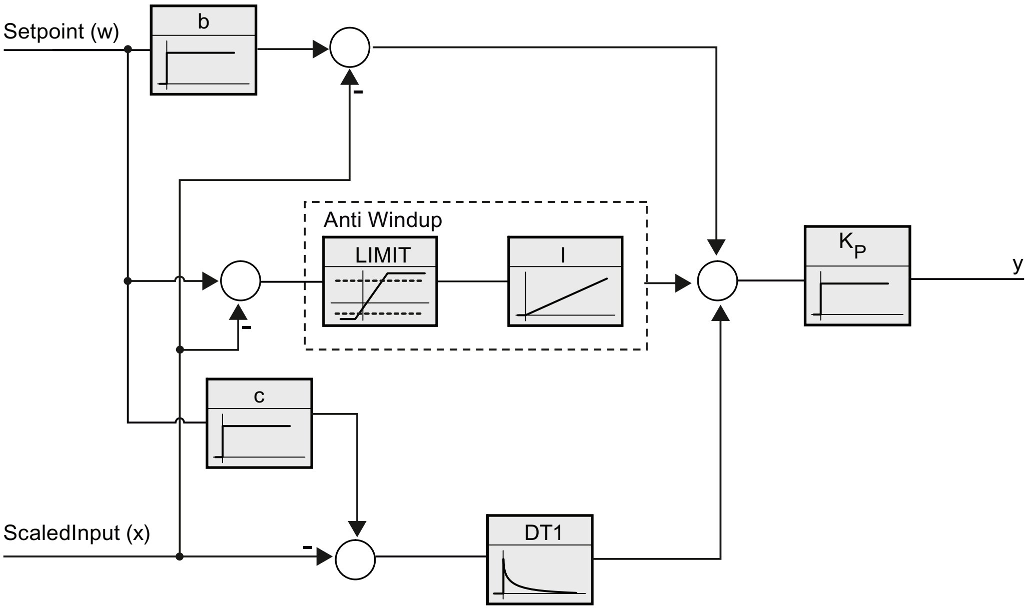
$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
y	Output value
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient (T1 = a × T_D)
	Derivative action time
c	Derivative action weighting

Block diagram of PID_Compact



Block diagram of PIDT1 with anti-windup



Reaction to errors

If errors occur, they are output in parameter Error, and PID_Compact changes to "Inactive" mode. Reset the errors using the Reset parameter.

See also

Controller type (Page 2889)

Input parameters of PID_Compact

Table 9- 52

Parameter	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A variable of the user program is used as source for the process value. If you are using parameter Input, then sPid_Cmpt.b_Input_PER_On = FALSE must be set.
Input_PER	WORD	W#16#0	Analog input as the source of the process value If you are using parameter Input_PER, then sPid_Cmpt.b_Input_PER_On = TRUE must be set.
ManualEnable	BOOL	FALSE	• A FALSE -> TRUE edge selects "Manual mode", while State = 4, sRet.i_Mode remains unchanged. • A TRUE -> FALSE edge selects the most recently active operating mode, State =sRet.i_Mode A change of sRet.i_Mode will not take effect during ManualEnable = TRUE. The change of sRet.i_Mode will only be considered upon a TRUE -> FALSE edge at ManualEnable . **PID_Compact V1.2 und PID_Compact V1.0** If at start of the CPU ManualEnable = TRUE, PID_Compact starts in manual mode. A rising edge (FALSE > TRUE) at ManualEnable is not necessary. **PID_Compact V1.1** At the start of the CPU, PID_Compact only switches to manual mode with a rising edge (FALSE->TRUE) at ManualEnable . Without rising edge, PID_Compact starts in the last operating mode in which ManualEnable was FALSE.
ManualValue	REAL	0.0	Manual value This value is used as the output value in manual mode.
Reset	BOOL	FALSE	The Reset parameter (Page 1785) restarts the controller.

Output parameters of PID_Compact

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Output of the scaled process value
Outputs "Output", "Output_PER", and "Output_PWM" can be used concurrently.			
Output	REAL	0.0	Output value in REAL format
Output_PER	WORD	W#16#0	Analog output value
Output_PWM	BOOL	FALSE	Pulse-width-modulated output value The output value is formed by minimum On and Off times.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the setpoint absolute high limit is reached. The setpoint in the CPU is limited to the configured setpoint absolute high limit. The configured process value absolute high limit is the default for the setpoint high limit. If you set sPid_Cmpt.r_Sp_Hlm to a value within the process value limits, this value is used as the setpoint high limit.
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the setpoint absolute low limit has been reached. In the CPU, the setpoint is limited to the configured setpoint absolute low limit. The configured process value absolute low limit is the default setting for the setpoint low limit. If you set sPid_Cmpt.r_Sp_Llm to a value within the process value limits, this value is used as the setpoint low limit.
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.
State	INT	0	The State parameter (Page 1781) shows the current operating mode of the PID controller. To change the operating mode, use variable sRet.i_Mode. • State = 0: Inactive • State = 1: pretuning • State = 2: fine tuning • State = 3: Automatic mode • State = 4: Manual mode
Error	DWORD	W#16#0	The Error parameter (Page 1784) indicates the error messages. Error = 0000: No error pending.

Static variables of PID_Compact

You must not modify variables that are not listed. These are used for internal purposes only.

Table 9- 53

Variable	Data type	Default	Description
sb_GetCycleTime	BOOL	TRUE	If sb_GetCycleTime = TRUE, the automatic determination of the cycle time is started. CycleTime.StartEstimation = FALSE once measurement is complete.
sb_EnCyclEstimation	BOOL	TRUE	If sb_EnCyclEstimation = TRUE, the sampling time PID_Compact is calculated.
sb_EnCyclMonitoring	BOOL	TRUE	If sb_EnCyclMonitoring = FALSE, the sampling time PID_Compact is not monitored. If it is not possible to execute PID_Compact within the sampling time, an 0800 error is not output and PID_Compact does not change to "Inactive" mode.
sb_RunModeByStartup	BOOL	TRUE	Enable most recent CPU mode upon CPU restart If sb_RunModeByStartup = FALSE, the controller will remain inactive after a CPU startup. After a CPU startup and if sb_RunModeByStartup = TRUE, the controller will return to the most recently active operating mode.
si_Unit	INT	0	Unit of measurement of the process value and setpoint, e.g., ºC, or ºF.
si_Type	INT	0	Physical quantity of the process value and setpoint, e.g., temperature.
sd_Warning	DWORD	DW#16#0	Variable sd_warning (Page 1786) displays the warnings generated since the reset, or since the last change of the operating mode.
sBackUp.r_Gain	REAL	1.0	Saved proportional gain You can reload values from the sBackUp structure with sPid_Cmpt.b_LoadBackUp = TRUE.
sBackUp.r_Ti	REAL	20.0	Stored integral action time [s]
sBackUp.r_Td	REAL	0.0	Stored derivative action time [s]
sBackUp.r_A	REAL	0.0	Saved derivative delay coefficient
sBackUp.r_B	REAL	0.0	Saved proportional action weighting factor
sBackUp.r_C	REAL	0.0	Saved derivative action weighting factor
sBackUp.r_Cycle	REAL	1.0	Saved Sampling time PID algorithm
sPid_Calc.r_Cycle	REAL	0.1	Sampling time of the PID_Compact instruction r_Cycle is determined automatically and usually equivalent to the cycle time of the calling OB.

Variable	Data type	Default	Description
sPid_Calc.b_RunIn	BOOL	FALSE	• b_RunIn = FALSE Pretuning is started when fine tuning is started in inactive or manual mode. If the requirements for pretuning are not met, PID_Compact reacts like b_RunIn = TRUE. The existing PID parameters are used to regulate the setpoint when fine tuning is started from automatic mode. Only then will fine tuning start. If pretuning is not possible, PID_Compact will change to "Inactive" mode. • b_RunIn = TRUE The pretuning is skipped. PID_3Compact tries to reach the setpoint with minimum or maximum output value. This can produce increased overshoot. Fine tuning will then start automatically. b_RunIn is set to FALSE after fine tuning.
sPid_Calc.b_CalcParamSUT	BOOL	FALSE	The parameters for pretuning will be recalculated if b_CalcParamSUT = TRUE. This enables you to change the parameter calculation method without having to repeat controller tuning. b_CalcParamSUT will be set to FALSE after calculation.
sPid_Calc.b_CalcParamTIR	BOOL	FALSE	The parameters for fine tuning will be recalculated if b_CalcParamTIR = TRUE. This enables you to change the parameter calculation method without having to repeat controller tuning.# b_CalcParamTIR will be set to FALSE after calculation.
sPid_Calc.i_CtrlTypeSUT	INT	0	Methods used to calculate parameters during pretuning: • i_CtrlTypeSUT = 0: PID according to Chien, Hrones and Restwick • i_CtrlTypeSUT = 1: PI according to Chien, Hrones and Restwick
sPid_Calc.i_CtrlTypeTIR	INT	0	Methods used to calculate parameters during fine tuning: • i_CtrlTypeTIR = 0: A PID automatically • i_CtrlTypeTIR = 1: A PID fast • i_CtrlTypeTIR = 2: A PID slow • i_CtrlTypeTIR = 3: Ziegler-Nichols PID • i_CtrlTypeTIR = 4: Ziegler-Nichols PI • i_CtrlTypeTIR = 5: Ziegler-Nichols P

Variable	Data type	Default	Description
sPid_Calc.r_Progress	REAL	0.0	Progress of controller tuning as a percentage (0.0 - 100.0)
sPid_Cmpt.r_Sp_Hlm	REAL	+3.402822e+38	High limit of setpoint If you set sPid_Cmpt.r_Sp_Hlm outside the process value limits, the configured process value absolute high limit is used as the setpoint high limit. If you set sPid_Cmpt.r_Sp_Hlm within the process value limits, this value is used as the setpoint high limit.
sPid_Cmpt.r_Sp_Llm	REAL	-3.402822e+38	Low limit of the setpoint If you set sPid_Cmpt.r_Sp_Llm outside the process value limits, the configured process value absolute low limit is used as the setpoint low limit. If you set sPid_Cmpt.r_Sp_Llm within the process value limits, this value is used as the setpoint low limit.
sPid_Cmpt.r_Pv_Norm_IN_1	REAL	0.0	Scaling Input_PER low Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Pv_Norm_IN_2	REAL	27648.0	Scaling Input_PER high Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Pv_Norm_OUT_1	REAL	0.0	Scaled low process value Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Pv_Norm_OUT_2	REAL	100.0	Scaled high process value Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Lmn_Hlm	REAL	100.0	Output value high limit for output parameter "Output"
sPid_Cmpt.r_Lmn_Llm	REAL	0.0	Low output value limit for output parameter "Output"
sPid_Cmpt.b_Input_PER_On	BOOL	TRUE	If b_Input_PER_On = TRUE, then parameter Input_PER is used. If b_Input_PER_On = FALSE, then parameter Input is used.
sPid_Cmpt.b_LoadBackUp	BOOL	FALSE	Activate the back-up parameter set. If an optimization has failed, you can reactivate the previous PID parameters by setting this bit.
sPid_Cmpt.b_InvCtrl	BOOL	FALSE	Invert control logic With b_InvCtrl = TRUE, a rising control deviation reduces the output value.

Variable	Data type	Default	Description
sPid_Cmpt.r_Lmn_Pwm_PPTm	REAL	0.0	The minimum ON time of the pulse width modulation in seconds is rounded to r_Lmn_Pwm_PPTm = r_Cycle or r_Lmn_Pwm_PPTm = n*r_Cycle
sPid_Cmpt.r_Lmn_Pwm_PBTm	REAL	0.0	The minimum OFF time of the pulse width modulation in seconds is rounded to r_Lmn_Pwm_PBTm = r_Cycle or r_Lmn_Pwm_PBTm = n*r_Cycle
sPid_Cmpt.r_Pv_Hlm	REAL	120.0	High limit of the process value At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer reported for a violation of the "High limit process value". Only a wire-break and a short-circuit are recognized and the PID_Compact switches to "Inactive" mode. r_Pv_Hlm > r_Pv_Llm
sPid_Cmpt.r_Pv_Llm	REAL	0.0	Low limit of the process value r_Pv_Llm < r_Pv_Hlm
sPid_Cmpt.r_Pv_HWrn	REAL	+3.402822e+38	Warning high limit of the process value If you set r_Pv_HWrn outside the process value limits, the configured process value absolute high limit is used as the warning high limit. If you set r_Pv_HWrn within the process value limits, this value is used as the warning high limit. r_Pv_HWrn > r_Pv_LWrn r_Pv_HWrn ≤ r_Pv_Hlm
sPid_Cmpt.r_Pv_LWrn	REAL	-3.402822e+38	Warning low limit of the process value If you set r_Pv_LWrn outside the process value limits, the configured process value absolute low limit is used as the warning low limit. If you set r_Pv_LWrn within the process value limits, this value is used as the warning low limit. r_Pv_LWrn < r_Pv_HWrn r_Pv_LWrn ≥ r_Pv_LWrn
sParamCalc.i_Event_SUT	INT	0	Variable i_Event_SUT (Page 1787) indicates the current phase of "pretuning":
sParamCalc.i_Event_TIR	INT	0	Variable i_Event_TIR (Page 1787) indicates the current phase of "fine tuning":
sRet.i_Mode	INT	0	The operating mode is changed edge-controlled. The following operating mode is enabled on a change to • i_Mode = 0: "Inactive" (controller stop) • i_Mode = 1: "Pretuning" mode • i_Mode = 2: "Fine tuning" mode • i_Mode = 3: "Automatic mode" • i_Mode = 4: "Manual mode"

Variable	Data type	Default	Description
sRet.r_Ctrl_Gain	REAL	1.0	Active proportional gain
sRet.r_Ctrl_Ti	REAL	20.0	• r_Ctrl_Ti > 0.0: active integral action time • r_Ctrl_Ti = 0.0: Integral action is disabled
sRet.r_Ctrl_Td	REAL	0.0	• r_Ctrl_Td > 0.0: Active derivative action time • r_Ctrl_Td = 0.0: Derivative action is disabled
sRet.r_Ctrl_A	REAL	0.0	Active derivative delay coefficient
sRet.r_Ctrl_B	REAL	0.0	Active proportional action weighting
sRet.r_Ctrl_C	REAL	0.0	Active derivative action weighting
sRet.r_Ctrl_Cycle	REAL	1.0	Active sampling time of the PID algorithm r_Ctrl_Cycle is calculated during controller tuning and rounded to an integer multiple of r_Cycle.

Note

Edit the variables listed in this table in "Inactive" mode to avoid malfunction of the PID controller. The "Inactive" mode is forced by setting variable "sRet.i_Mode" to "0".

Parameters State and sRet.i_Mode

Correlation of the parameters

The State parameter indicates the current operating mode of the PID controller. You cannot modify the State parameter.

You need to modify the sRet.i_Mode tag to change the operating mode. This also applies when the value for the new operating mode is already in sRet.i_Mode. First set sRet.i_Mode = 0 and then sRet.i_Mode = 3. Provided the current operating mode of the controller supports this change, State is set to the value of sRet.i_Mode.

When PID_Compact automatically switches the operating mode, the following applies: State != sRet.i_Mode.

Examples:

- Successful pretuning
  State = 3 and sRet.i_Mode = 1
- Error
  State = 0 and sRet.i_Mode remains at the same value, e.g sRet.i_Mode = 3
- ManualEnalbe = TRUE
  State = 4 and sRet.i_Mode remain at the previous value, for example, sRet.i_Mode = 3

Note

You wish to repeat successful fine tuning without exiting automatic mode with i_Mode = 0.

Setting sRet.i_Mode to an invalid value such as 9999 for one cycle has no effect on State. Set Mode = 2 in the next cycle. You can generate a change to sRet.i_Mode without first switching to "inactive" mode.

Meaning of values

State / sRet.i_Mode	Description of the operating mode
0	Inactive The controller is switched off. The controller was in "inactive" mode before pretuning was performed. The PID controller will change to "inactive" mode when running if an error occurs or if the "Deactivate controller" icon is clicked in the commissioning window.
1	Pretuning The pretuning determines the process response to a jump of the output value and searches for the point of inflection. The optimized PID parameters are calculated as a function of the maximum rate of rise and dead time of the controlled system. Pretuning requirements: • The controller is in inactive mode or manual mode • ManualEnable = FALSE • The process value must not be too close to the setpoint. |Setpoint - Input| > 0.3 * |sPid_Cmpt.r_Pv_Hlm - sPid_Cmpt.r_Pv_Llm| and |Setpoint - Input| > 0.5 * |Setpoint| • The setpoint may not be changed during pretuning. The higher the stability of the process value, the easier it is to calculate the PID parameters and increase precision of the result. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. PID parameters are backed up before they are recalculated and can be reactivated with sPid_Cmpt.b_LoadBackUp. There is a change to automatic mode following successful pretuning and to "inactive" mode following unsuccessful pretuning. The phase of pretuning is indicated with Variable i_Event_SUT (Page 1787).

State / sRet.i_Mode	Description of the operating mode
2	Fine tuning Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are optimized based on the amplitude and frequency of this oscillation. The differences between the process response during pretuning and fine tuning are analyzed. All PID parameters are recalculated on the basis of the findings. PID parameters from fine tuning usually have better master control and disturbance behavior than PID parameters from pretuning. PID_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. PID parameters are backed up before they are recalculated and can be reactivated with sPid_Cmpt.b_LoadBackUp. Requirements for fine tuning: • No disturbances are expected. • The setpoint and the process value lie within the configured limits. • The setpoint may not be changed during fine tuning. • ManualEnable = FALSE • Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode Fine tuning proceeds as follows when started in: • Automatic mode (State = 3) Start fine tuning in automatic mode if you wish to improve the existing PID parameters using controller tuning. PID_Comact will regulate using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. • Inactive (State = 0) or manual (State = 4) mode If the requirements for pretuning are met, pretuning is started. The PID parameters established will be used for adjustment until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. If pretuning is not possible, PID_Compact will change to "Inactive" mode. An attempt is made to reach the setpoint with a minimum or maximum output value if the process value for pretuning is already too near the setpoint or sPid_Calc.b_RunIn = TRUE. This can produce increased overshoot. The controller will change to "automatic mode" after successfully completed "fine tuning" and to "inactive" mode if "fine tuning" has not been successfully completed. The "Fine tuning" phase is indicated with Variable i_Event_TIR (Page 1787).

State / sRet.i_Mode	Description of the operating mode
3	Automatic mode In automatic mode, PID_Compact corrects the controlled system in accordance with the parameters specified. The controller changes to automatic mode if one the following conditions is fulfilled: • Pretuning successfully completed • Fine tuning successfully completed • Change of variable sRet.i_Mode to the value 3. After CPU startup or change from Stop to RUN mode, PID_Compact will start in the most recently active operating mode. To retain PID_Compact in "Inactive" mode, set sb_RunModeByStartup = FALSE.
4	Manual mode In manual mode, you specify a manual output value in the ManualValue parameter. This operating mode is enabled if sRet.i_Mode = 4, or at the rising edge on ManualEnable. If ManualEnable changes to TRUE, only State will change. sRet.i_Mode will retain its current value. PID_Compact will return to the previous operating mode upon a falling edge at ManualEnable. The change to automatic mode is bumpless.

See also

Output parameters of PID_Compact (Page 1776)

Pretuning (Page 2898)

Fine tuning (Page 2899)

"Manual" mode (Page 2904)

Variable i_Event_SUT (Page 1787)

Variable i_Event_TIR (Page 1787)

Error parameter

If several errors are pending, the values of the error codes are displayed by means of binary addition. The display of error code 0003, for example, indicates that the errors 0001 and 0002 are also pending.

Error (DW#16#...)	Description
0000	There is no error.
0001	The "Input" parameter is outside the process value limits. • Input > sPid_Cmpt.r_Pv_Hlmor • Input < sPid_Cmpt.r_Pv_Llm You cannot start the actuator again until you eliminate the error.
0002	Invalid value at parameter "Input_PER". Check whether an error is pending at the analog input.
0004	Error during fine tuning Oscillation of the process value could not be maintained.
0008	Error while starting pretuning. The process value is too close to the setpoint. Start fine tuning.
0010	The setpoint was changed during controller tuning.

Error (DW#16#...)	Description
0020	Pretuning may not be carried out in automatic mode or during fine tuning.
0040	Error in fine tuning The setpoint is too close to the setpoint limits.
0080	Incorrect configuration of output value limits. Check to see if the limits of the output value are configured correctly and match the direction in which the control is operating.
0100	Error during controller tuning has resulted in invalid parameters.
0200	Invalid value at parameter "Input": Numerical format of value is invalid.
0400	Calculating the output value failed. Check the PID parameters.
0800	Sampling time error: PID_Compact is not called within the sampling time of the cyclic interrupt OB.
1000	Invalid value at parameter "Setpoint": Numerical format of value is invalid.

See also

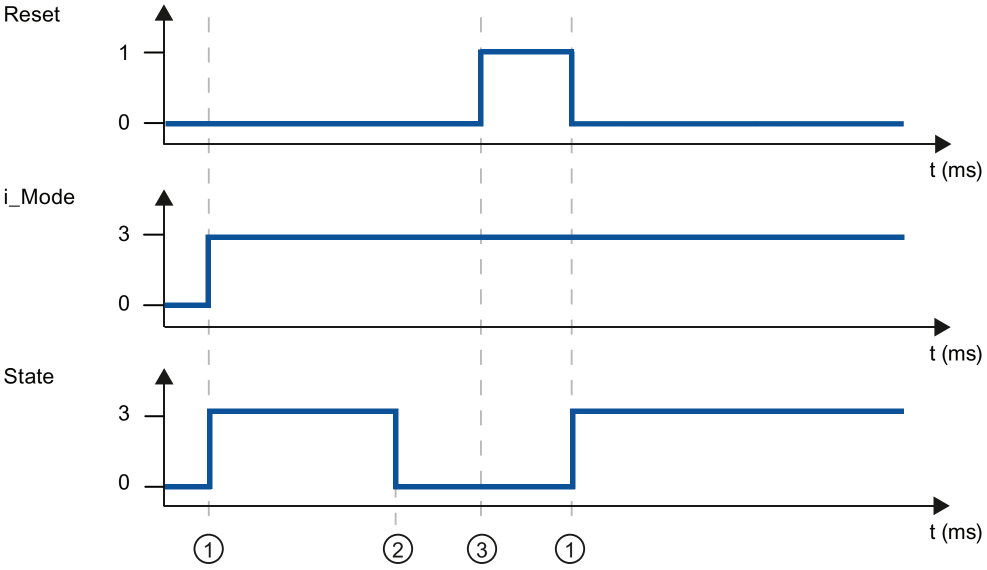
Output parameters of PID_Compact (Page 1776)

Reset parameter

The response to Reset = TRUE depends on the version of the PID_Compact instruction.

Reset response PID_Compact V.1.1

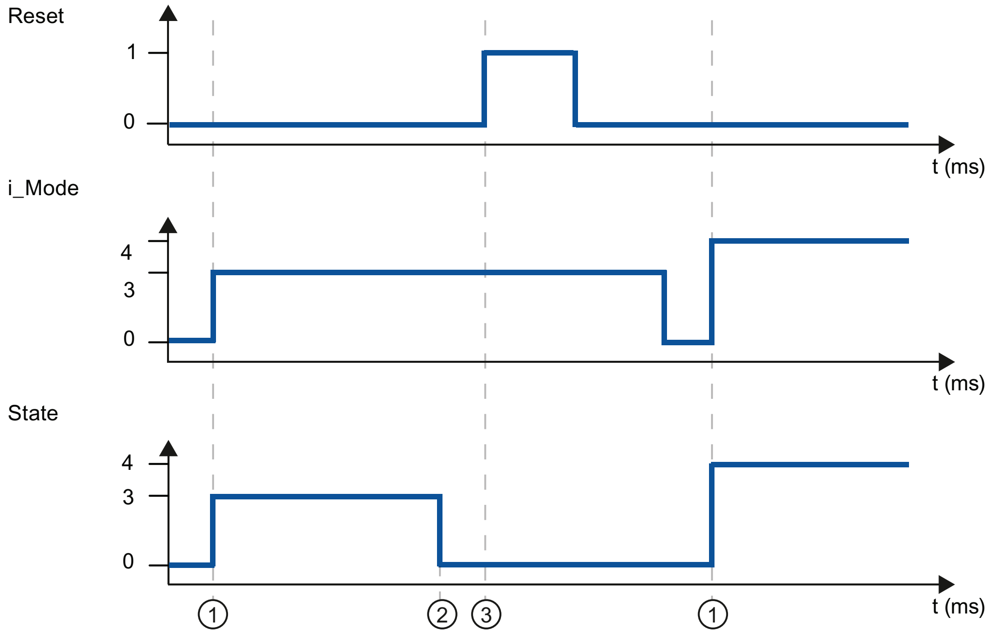
A rising edge at Reset resets the errors and warnings and clears the integral action. A falling edge at Reset triggers a change to the most recently active operating mode.



① Activation
② Error
③ Reset

Reset response PID_Compact V.1.0

A rising edge at Reset resets the errors and warnings and clears the integral action. The controller is not reactivated until the next edge at i_Mode.



① Activation

② Error

③ Reset

Variable sd_warning

If several warnings are pending, the values of variable sd_warning are displayed by means of binary addition. The display of warning 0003, for example, indicates that the warnings 0001 and 0002 are also pending.

sd_warning (DW#16#....)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0002	Oscillation increased during fine tuning.
0004	The setpoint was outside the set limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. The PID parameters were instead calculated using the "i_CtrlTypeTIR = 3" method.
0010	The operating mode could not be changed because ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.

The following warnings are deleted as soon as the cause is dealt with:

- 0004
- 0020
- 0040

All other warnings are cleared with a rising edge at Reset.

Variable i_Event_SUT

i_Event_SUT	Name	Description
0	SUT_INIT	Initialize pretuning
100	SUT_STDABW	Calculate the standard deviation
200	SUT_GET_POI	Find the point of inflection
9900	SUT_IO	Pretuning successful
1	SUT_NIO	Pretuning not successful

See also

Static variables of PID_Compact (Page 1777)

Parameters State and sRet.i_Mode (Page 1781)

Variable i_Event_TIR

i_Event_TIR	Name	Description
-100	TIR_FIRST_SUT	Fine tuning is not possible. Pretuning will be executed first.
0	TIR_INIT	Initialize fine tuning
200	TIR_STDABW	Calculate the standard deviation
300	TIR_RUN_IN	Attempt to reach the setpoint
400	TIR_CTRLN	Attempt to reach the setpoint with the existing PID parameters (if pretuning has been successful)
500	TIR_OSZIL	Determine oscillation and calculate parameters
9900	TIR_IO	Fine tuning successful
1	TIR_NIO	Fine tuning not successful

See also

Static variables of PID_Compact (Page 1777)

Parameters State and sRet.i_Mode (Page 1781)

PID_3Step

Description of PID_3Step

Description

Use the PID_3Step instruction to configure a PID controller with integrated tuning for valves or actuators with integral action.

The following operating modes are possible:

- Inactive
- Pretuning
- Fine tuning
- Automatic mode
- Manual mode
- Approach substitute output value
- Transition time measurement
- Approach substitute output value with error monitoring
- Error monitoring

For more information on operating modes, see the State parameter.

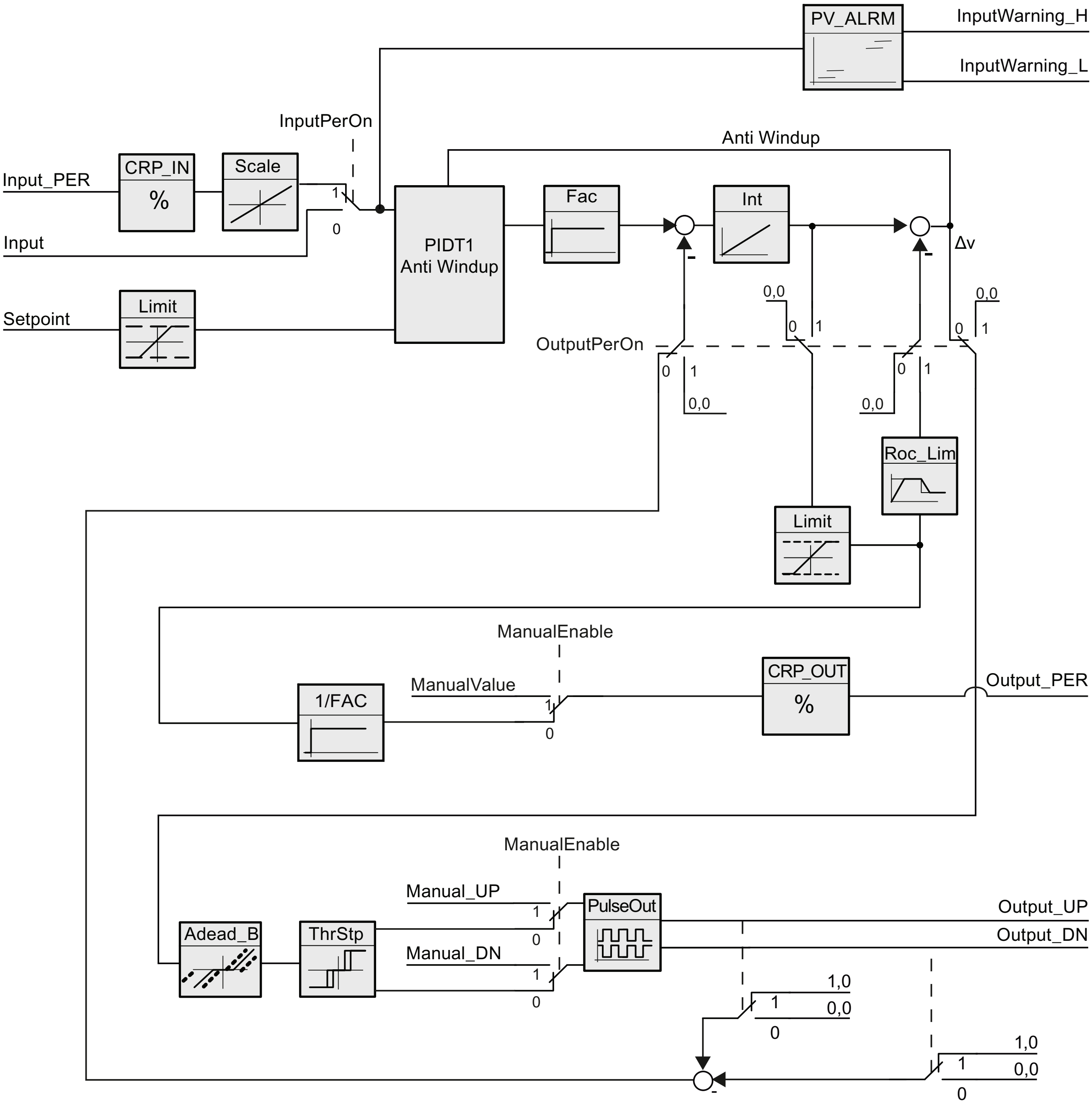
PID algorithm

PID_3Step is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The following equation is used to calculate the output value.

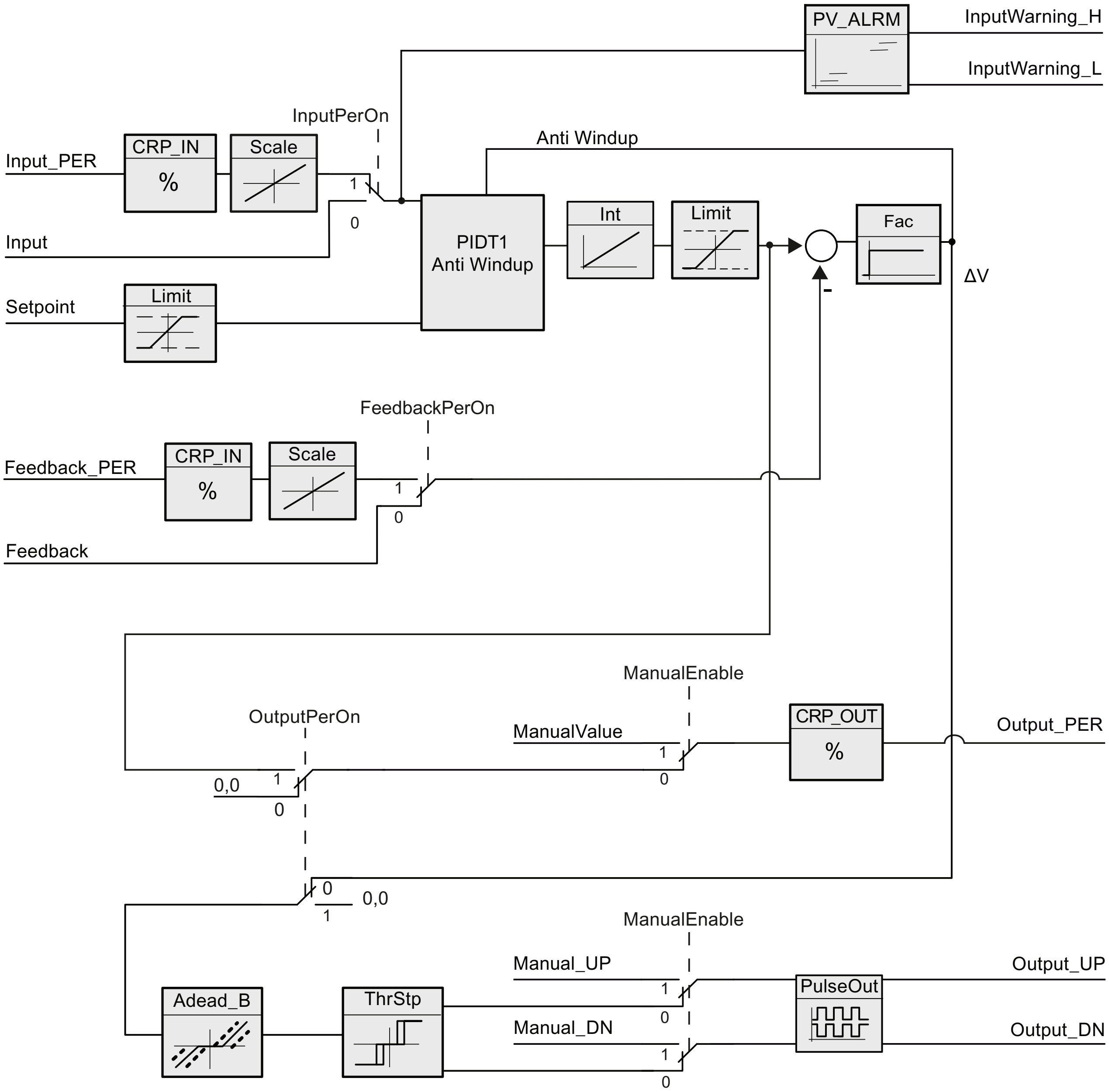
$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
y	Output value
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient ($T1 = a \times T_D$)
T_D	Derivative action time
c	Derivative action weighting

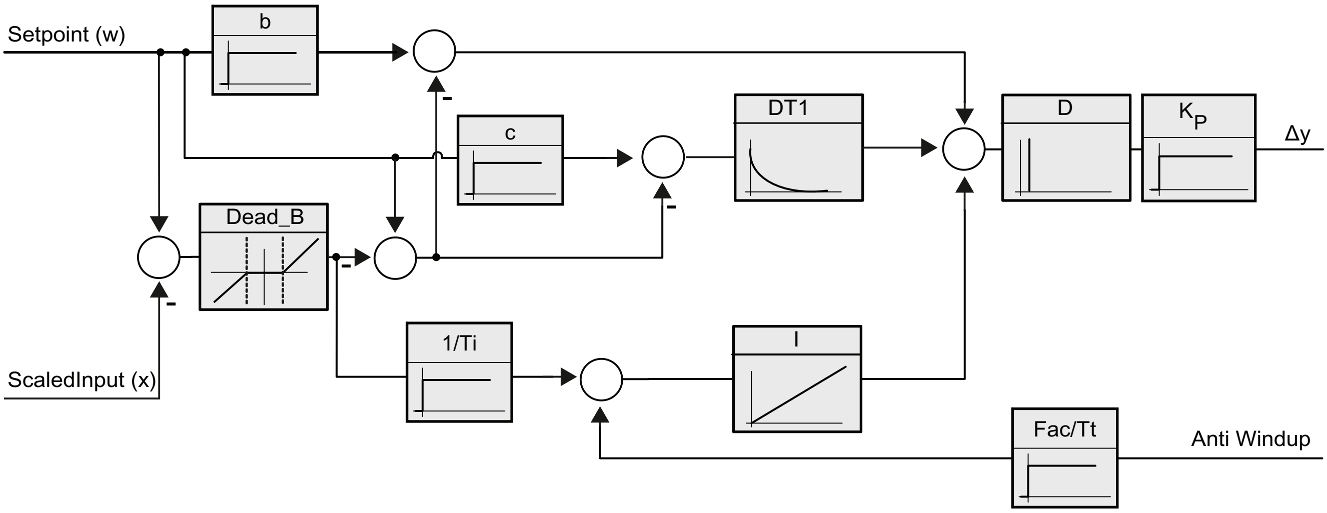
Block diagram without position feedback



Block diagram with position feedback



Block diagram of PIDT1 with anti-windup



Call

PID_3Step is called in the constant interval of the cycle time of the calling OB (preferably in a cyclic interrupt OB).

Startup

At the startup of the CPU, PID_3Step starts in the operating mode that was last active. To retain PID_3Step in "Inactive" mode, set RunModeByStartup = FALSE.

Reaction to errors

If errors occur, these are output in the Error parameter. Configure the response of PID_3Step using the ErrorBehaviour and ActivateRecoverMode tags.

ErrorBehaviour	ActivateRecoverMode	Actuator setting configuration Set Output to	Response
0	FALSE	Current output value	Change to "Inactive" mode (Mode = 0)
0	TRUE	Current output value while error is pending	Change to "Error monitoring" mode (Mode = 8)
1	FALSE	Substitute output value	Change to "Approach substitute output value" mode (Mode = 5) Change to "Inactive" mode (Mode = 0)
1	TRUE	Substitute output value while error is pending	Change to "Approach substitute output value with error monitoring" mode (Mode = 7) Change to "Error monitoring" mode (Mode = 8)

The ErrorBits parameter identifies the errors.

See also

State and Retain.Mode parameters (Page 1804)

ErrorBits parameter (Page 1811)

Operating principle of PID_3Step

Monitoring process value limits

Specify the absolute high and low limit of the process value in the Config.InputUpperLimit and Config.InputLowerLimit tags. Process values outside these limits trigger an error (ErrorBits = 0001hex).

Specify the absolute high and low warning limit in the Config.InputUpperWarning and Config.InputLowerWarning tags. Process values outside these warning limits trigger a warning (Warnings = 0040hex) and the output parameter InputWarning_H or InputWarning_L changes to TRUE.

The high and low limits of the process value are set as default warning limits.

As an exception, the process value limits are **not** monitored in the "Inactive" operating mode and during reset.

Limiting the setpoint

Use the Config.SetpointUpperLimit and Config.SetpointLowerLimit tags to specify a high and low limit of the setpoint. PID_3Step automatically limits the setpoint to the process value limits. You can limit the setpoint to a smaller range. PID_3Step checks whether this range falls within the process value limits. Setpoints outside these limits are replaced with the high or low limit, and output parameter SetpointLimit_H or SetpointLimit_L will be set to TRUE.

The setpoint is limited in all operating modes.

Monitor output value limits

Specify the absolute high and low limit of the output value in the Config.OutputUpperLimit and Config.OutputLowerLimit tags. The output value limits must be within "Low endstop" and "High endstop".

- High endstop: Config.FeedbackScaling.UpperPointOut
- Low endstop: Config.FeedbackScaling.LowerPointOut

Rule:

UpperPointOut ≥ OutputUpperLimit > OutputLowerLimit ≤ LowerPointOut

The valid values for "High endstop" and "Low endstop" depend upon:

- FeedbackOn
- FeedbackPerOn
- OutputPerOn

OutputPerOn	FeedbackOn	FeedbackPerOn	LowerPointOut	UpperPointOut
FALSE	FALSE	FALSE	0.0% cannot be set	100.0% cannot be set
FALSE	TRUE	FALSE	-100.0 % or 0.0 %	0.0% or +100.0 %
FALSE	TRUE	TRUE	-100.0 % or 0.0 %	0.0% or +100.0 %
TRUE	FALSE	FALSE	0.0% cannot be set	100.0% cannot be set
TRUE	TRUE	FALSE	-100.0 % or 0.0 %	0.0% or +100.0 %
TRUE	TRUE	TRUE	-100.0 % or 0.0 %	0.0% or +100.0 %

If OutputPerOn = FALSE and FeedbackOn = FALSE, you cannot limit the output value. The digital outputs are reset with Actuator_H = TRUE or Actuator_L = TRUE, or after a travel time amounting to 110% of the motor transition time.

The output value is 27648 at 100% and -27648 at -100%. PID_3Step must be able to close the valve completely. Therefore, zero must be included in the output value limits.

Substitute output value

If an error has occurred, PID_3Step can output a substitute output value and set the actuator to a safe position that is specified at the SavePosition variable. The substitute output value must be within the output value limits.

Monitoring signal validity

The validity of the Setpoint, Input, and Output parameter values is monitored.

Monitoring the sampling time PID_3Step

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID_3Step instruction measures the time interval between two calls. This is the current sampling time. On every change of the operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. Too great a difference between the current sampling time and this mean value triggers an error (ErrorBits = 0800 hex).

PID_3Step is set to "Inactive" mode during controller tuning under the following conditions:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value

In automatic mode, PID_3Step is set to "Inactive" mode under the following conditions:

- New mean value >= 1.5 x old mean value
- New mean value <= 0.5 x old mean value

Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during controller tuning and rounded to a multiple of the cycle time. All other functions of PID_3Step are executed at every call.

Measuring the motor transition time

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. The actuator is moved for a maximum of 110% of the motor transition time in one direction. PID_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation are average values for this type of actuator. The value for the specific actuator used may differ. You can measure the motor transition time during commissioning.

Effective direction

For cooling and discharge control systems, it may be necessary to invert the effective direction. This inversion is controlled using variable Config.InvertControl.

Input parameters of PID_3Step

Table 9- 54

Parameters	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A variable of the user program is used as source for the process value. If you are using parameter Input, then Config.InputPerOn = FALSE must be set.
Input_PER	WORD	W#16#0	An analog input is used as source for the process value. If you are using parameter Input_PER, then Config.InputPerOn = TRUE must be set.
Actuator_H	BOOL	FALSE	Digital position feedback of the valve for the high endstop If Actuator_H = TRUE, the valve is at the high endstop and is no longer moved towards this direction.
Actuator_L	BOOL	FALSE	Digital position feedback of the valve for the low endstop If Actuator_L = TRUE, the valve is at the low endstop and is no longer moved towards this direction.
Feedback	REAL	0.0	Position feedback of the valve If you are using parameter Feedback, then Config.FeedbackPerOn = FALSE must be set.

Parameters	Data type	Default	Description
Feedback_PER	WORD	W#16#0	Analog feedback of the valve position If you are using parameter Feedback_PER, then Config.FeedbackPerOn = TRUE must be set. Feedback_PER is scaled based on the variables: • Config.FeedbackScaling.LowerPointIn • Config.FeedbackScaling.UpperPointIn • Config.FeedbackScaling.LowerPointOut • Config.FeedbackScaling.UpperPointOut
ManualEnable	BOOL	FALSE	• A FALSE -> TRUE edge selects "Manual mode", while State = 4, Retain.Mode remains unchanged. • A TRUE -> FALSE edge selects the most recently active operating mode A change of Retain.Mode will not take effect during ManualEnable = TRUE. The change of Retain.Mode will only be considered upon a TRUE -> FALSE edge at ManualEnable . **PID_3Step V1.1**If at start of the CPU ManualEnable = TRUE, PID_3Step starts in manual mode. A rising edge (FALSE > TRUE) at ManualEnable is not necessary. **PID_3Step V1.0** At the start of the CPU, PID_3Step only switches to manual mode with a rising edge (FALSE->TRUE) at ManualEnable . Without rising edge, PID_3Step starts in the last operating mode in which ManualEnable was FALSE.
ManualValue	REAL	0.0	In manual mode, you specify the absolute position of the valve. ManualValue will only be evaluated if you are using OutputPer, **or** if position feedback is available.
Manual_UP	BOOL	FALSE	In manual mode, every rising edge opens the valve by 5% of the total control range, or for the duration of the minimum motor transition time. Manual_UP is evaluated only if you are not using Output_PER and there is no position feedback available.
Manual_DN	BOOL	FALSE	In manual mode, every rising edge closes the valve by 5% of the total control range, or for the duration of the minimum motor transition time. Manual_DN is evaluated only if you are not using Output_PER and there is no position feedback available.
Reset	BOOL	FALSE	Restarts the controller. • FALSE -> TRUE edge – Change to "Inactive" mode – Intermediate controller values are reset (PID parameters are retained) • TRUE -> FALSE edge Change in most recent active mode

Output parameters of PID_3Step

Parameters	Data type	Default	Description
ScaledInput	REAL	0.0	Scaled process value
ScaledFeedback	REAL	0.0	Scaled position feedback
Output_UP	BOOL	FALSE	Digital output value for opening the valve If Config.OutputPerOn = FALSE, then parameter Output_UP is used.
Output_DN	BOOL	FALSE	Digital output value for closing the valve If Config.OutputPerOn = FALSE, then parameter Output_DN is used.
Output_PER	WORD	W#16#0	Analog output value If Config.OutputPerOn = TRUE, Output_PER is used.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the setpoint absolute high limit is reached. The setpoint in the CPU is limited to the configured setpoint absolute high limit. The configured process value absolute high limit becomes the default for the setpoint high limit. If you set Config.SetpointUpperLimit to a value within the process value limits, this value is used as the setpoint high limit.
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the setpoint absolute low limit has been reached. In the CPU, the setpoint is limited to the configured setpoint absolute low limit. The configured process value absolute low limit is the default setting for the setpoint low limit. If you set Config.SetpointLowerLimit to a value within the process value limits, this value is used as the setpoint low limit.
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.
State	INT	0	The State parameter (Page 1804) shows the current operating mode of the PID controller. To change the operating mode, use variable Retain.Mode. • State = 0: Inactive • State = 1: pretuning • State = 2: fine tuning • State = 3: Automatic mode • State = 4: Manual mode • State = 5: substitute output value approach • State = 6: transition time measurement • State = 7: substitute output value approach with error monitoring • State = 8: Error monitoring

Parameters	Data type	Default	Description
Error	BOOL	FALSE	If Error = TRUE, at least one error message is pending.
ErrorBits	DWORD	DW#16#0	The ErrorBits parameter (Page 1811) indicates the error messages.

See also

State and Retain.Mode parameters (Page 1804)

ErrorBits parameter (Page 1811)

PID_3Step static variables

You must not modify variables that are not listed. These are used for internal purposes only.

Table 9- 55

Variable	Data type	Default	Description
ActivateRecoverMode	BOOL	TRUE	The ActivateRecoverMode variable (Page 1813) determines the behavior in the event of an error.
RunModeByStartup	BOOL	TRUE	Enable most recent CPU mode upon CPU restart After a CPU startup and if RunModeByStartup = TRUE, the controller will return to the most recently active operating mode. If RunModeByStartup = FALSE, the controller will remain inactive after a CPU startup.
PhysicalUnit	INT	0	Unit of measurement of the process value and setpoint, e.g., ºC, or ºF.
PhysicalQuantitiy	INT	0	Physical quantity of the process value and setpoint, e.g., temperature.
ErrorBehaviour	INT	0	If ErrorBehaviour = 0 and an error has occurred, the valve stays at its current position and the controller changes directly to "Inactive" or "Error monitoring" mode. If ErrorBehaviour = 1 and an error occurs, the actuator moves to the substitute output value and only then switches to "Inactive" or "Error monitoring" mode. If the following errors occur, you can no longer move the valve to a configured substitute output value. • 2000h: Invalid value at parameter Feedback_PER. • 4000h: Invalid value at parameter Feedback. • 8000h: Error in digital position feedback.
Warnings	DWORD	DW#16#0	Variable Warnings (Page 1804) displays the warnings generated since the reset, or since the last change of the operating mode. Warnings: Cyclic warnings (e.g. process value warnings) are shown until the cause of the warning is removed. They are automatically deleted once their cause has gone. Non-cyclic warnings (e.g. point of inflection not found) remain and are deleted like errors.

Variable	Data type	Default	Description
SavePosition	REAL	0.0	Substitute output value If ErrorBehaviour = 1 and an error occurs, the actuator moves to a safe position for the plant and only then switches to "Inactive" mode.
CurrentSetpoint	REAL	0.0	Currently active setpoint This value is frozen during controller tuning.
Progress	REAL	0.0	Progress of controller tuning as a percentage (0.0 - 100.0)
Config.InputPerOn	BOOL	TRUE	If InputPerOn = TRUE, then parameter Input_PER is used. If InputPerOn = FALSE, then parameter Input is used.
Config.OutputPerOn	BOOL	FALSE	If OutputPerOn = TRUE, then parameter Output_PER is used. If OutputPerOn = FALSE, the Ouput_UP and Output_DN parameters will be used.
Config.LoadBackUp	BOOL	FALSE	If LoadBackUp = TRUE, the last set of PID parameters is reloaded. This set was saved prior to the last controller tuning operation.
Config.InvertControl	BOOL	FALSE	Invert control logic With InvertControl = TRUE, a rising control deviation reduces the output value.
Config.FeedbackOn	BOOL	FALSE	If FeedbackOn = FALSE, then a position feedback is simulated. Position feedback is always enabled if FeedbackOn = TRUE.
Config.FeedbackPerOn	BOOL	FALSE	FeedbackPerOn is only effective when FeedbackOn = TRUE. The analog input will be used for position feedback if FeedbackPerOn = TRUE (parameter Feedback_PER). The parameter Feedback will be used for position feedback if FeedbackPerOn = FALSE.
Config.ActuatorEndStopOn	BOOL	FALSE	If ActuatorEndStopOn = TRUE, the digital position feedback Actuator_L and Actuator_H are taken into consideration.
Config.InputUpperLimit	REAL	120.0	High limit of the process value At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer reported for a violation of the "High limit process value". Only a wire-break and a short-circuit are recognized and the PID_3Step behaves as configured in the event of an error. InputUpperLimit > InputLowerLimit
Config.InputLowerLimit	REAL	0.0	Low limit of the process value InputLowerLimit < InputUpperLimit
Config.InputUpperWarning	REAL	+3.402822e+38	Warning high limit of the process value If you set InputUpperWarning outside the process value limits, the configured process value absolute high limit is used as the warning high limit. If you set InputUpperWarning within the process value limits, this value is used as the warning high limit. InputUpperWarning > InputLowerWarning InputUpperWarning ≤ InputUpperLimit

Variable	Data type	Default	Description
Config.InputLowerWarning	REAL	-3.402822e+38	Warning low limit of the process value If you set InputLowerWarning outside the process value limits, the configured process value absolute low limit is used as the warning low limit. If you set InputLowerWarning within the process value limits, this value is used as the warning low limit. InputLowerWarning < InputUpperWarning InputLowerWarning ≥ InputLowerLimit
Config.OutputUpperLimit	REAL	100.0	High limit of output value For details, see OutputLowerLimit
Config.OutputLowerLimit	REAL	0.0	Low limit of output value If OutputPerOn = TRUE or FeedbackOn = TRUE, the range of values from -100% to +100% is valid, including zero. At -100%, Output = -27648; at +100% Output = 27648 If OutputPerOn = FALSE, the range of values from 0% to 100% is valid. The valve is completely closed at 0% and completely opened at 100%.
Config.SetpointUpperLimit	REAL	+3.402822e+38	High limit of setpoint If you set SetpointUpperLimit outside the process value limits, the configured process value absolute high limit defaults to the setpoint high limit. If you set SetpointUpperLimit within the process value limits, this value is used as the setpoint high limit.
Config.SetpointLowerLimit	REAL	-3.402822e+38	Low limit of the setpoint If you set SetpointLowerLimit outside the process value limits, the configured process value absolute low limit defaults to the setpoint low limit. If you set SetpointLowerLimit within the process value limits, this value is used as the setpoint low limit.
Config.MinimumOnTime	REAL	0.0	Minimum ON time Minimum time in seconds for which the servo drive must be switched on.
Config.MinimumOffTime	REAL	0.0	Minimum OFF time Minimum time in seconds for which the servo drive must be switched off.
Config.TransitTime	REAL	30.0	Motor transition time Time in seconds the actuating drive requires to move the valve from the closed to the opened state.
Config.InputScaling.UpperPointIn	REAL	27648.0	Scaling Input_PER high Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn from the InputScaling structure.
Config.InputScaling.LowerPointIn	REAL	0.0	Scaling Input_PER low Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn from the InputScaling structure.

Variable	Data type	Default	Description
Config.InputScaling.UpperPointOut	REAL	100.0	Scaled high process value Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn from the InputScaling structure.
Config.InputScaling.LowerPointOut	REAL	0.0	Scaled low process value Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn from the InputScaling structure.
Config.FeedbackScaling.UpperPointIn	REAL	27648.0	Scaling Feedback_PER high Feedback_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn from the FeedbackScaling structure.
Config.FeedbackScaling.LowerPointIn	REAL	0.0	Scaling Feedback_PER low Feedback_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn from the FeedbackScaling structure.
Config.FeedbackScaling.UpperPointOut	REAL	100.0	High endstop Feedback_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn from the FeedbackScaling structure.
Config.FeedbackScaling.LowerPointOut	REAL	0.0	Low endstop Feedback_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn from the FeedbackScaling structure.
GetTransitTime.InvertDirection	BOOL	FALSE	If InvertDirection = FALSE, the valve is fully opened, closed, and then reopened in order to determine the valve transition time. If InvertDirection = TRUE, the valve is fully closed, opened, and then closed again.
GetTransitTime.SelectFeedback	BOOL	FALSE	If SelectFeedback = TRUE, then Feedback_PER or Feedback will be included in the transition time measurement. If SelectFeedback = FALSE, then Actuator_H or Actuator_L will be included in the transition time measurement.
GetTransitTime.Start	BOOL	FALSE	Transition time measurement will be started if Start = TRUE.
GetTransitTime.State	INT	0	Current phase of the transition time measurement • State = 0: Inactive • State = 1: Open valve completely • State = 2: Close valve completely • State = 3: Set the valve to target position (NewOutput) • State = 4: Transition time measurement was successfully completed • State = 5: Transition time measurement was canceled
GetTransitTime.NewOutput	REAL	0.0	Target position for transition time measurement with position feedback NewOutput must be within the output value limits of the valve. The difference between NewOutput and ScaledFeedback must be at least 50% of the valid contgrol range.

Variable	Data type	Default	Description
CycleTime.StartEstimation	BOOL	TRUE	Measurement of the PID_3Step sampling time will start if StartEstimation = TRUE. CycleTime.StartEstimation = FALSE once measurement is complete.
CycleTime.EnEstimation	BOOL	TRUE	The sampling time PID_3Step will be calculated if EnEstimation = TRUE.
CycleTime.EnMonitoring	BOOL	TRUE	The sampling time PID_3Step will be monitored if EnMonitoring = TRUE. If it is not possible to execute PID_3Step within the sampling time, the error 0800 will be output and the operating mode changed. ActivateRecoverMode and ErrorBehaviour determine which operating mode is changed to. If EnMonitoring = FALSE, the sampling time PID_3Step will not be monitored, the error 0800 will not be output and the operating mode will not be changed.
CycleTime.Value	REAL	0.1	Sampling time PID_3Step CycleTime.Value is determined automatically and is usually equivalent to the cycle time of the calling OB.
CtrlParamsBackUp.SetByUser	BOOL	FALSE	Saved value of Retain.CtrlParams.SetByUser. You can reload values from the CtrlParamsBackUp structure with Config.LoadBackUp = TRUE.
CtrlParamsBackUp.Gain	REAL	1.0	Saved proportional gain
CtrlParamsBackUp.Ti	REAL	20.0	Saved integral action time
CtrlParamsBackUp.Td	REAL	0.0	Saved derivative action time
CtrlParamsBackUp.TdFiltRatio	REAL	0.0	Saved derivative delay coefficient
CtrlParamsBackUp.PWeighting	REAL	0.0	Saved proportional action weighting
CtrlParamsBackUp.DWeighting	REAL	0.0	Saved derivative action weighting
CtrlParamsBackUp.Cycle	REAL	1.0	Saved Sampling time PID algorithm
CtrlParamsBackUp.InputDeadBand	REAL	0.0	Saved dead band width of the control deviation
PIDSelfTune.SUT.CalculateSUTParams	BOOL	FALSE	The properties of the controlled system will be saved during controller tuning. The PID parameters will be recalculated on the basis of these properties if CalculateSUTParams = TRUE. The PID parameters will be calculated using the method set in TuneRuleSUT. CalculateSUTParams will be set to FALSE following calculation.
PIDSelfTune.SUT.TuneRuleSUT	INT	1	Methods used to calculate parameters during pretuning: • TuneRuleSUT = 0: Chien, Hrones and Reswick PID • TuneRuleSUT = 1: Chien, Hrones, Reswick PI
PIDSelfTune.SUT.State	INT	0	Variable SUT.State indicates the current phase of pretuning:

Variable	Data type	Default	Description
PIDSelfTune.TIR.RunIn	BOOL	FALSE	- RunIn = FALSE Pretuning is started when fine tuning is started in inactive or manual mode. The existing PID parameters are used to regulate the setpoint when fine tuning is started from automatic mode. Only then will fine tuning start. If pretuning is not possible, PID_3Step will change to "Inactive" mode. - RunIn = TRUE The pretuning is skipped. PID_3Step tries to reach the setpoint with minimum or maximum output value. This can produce increased overshoot. Only then will fine tuning start. RunIn is set to FALSE after fine tuning.
PIDSelfTune.TIR.CalculateTIR Params	BOOL	FALSE	The properties of the controlled system will be saved during controller tuning. The PID parameters will be recalculated on the basis of these properties if CalculateTIRParams = TRUE. The PID parameters will be calculated using the method set in TuneRuleTIR. CalculateTIRParams will be set to FALSE following calculation.
PIDSelfTune.TIR.TuneRuleTI R	INT	0	Methods used to calculate parameters during fine tuning: - TuneRuleTIR = 0: PID automatic - TuneRuleTIR = 1: PID rapid - TuneRuleTIR = 2: PID slow - TuneRuleTIR = 3: Ziegler-Nichols PID - TuneRuleTIR = 4: Ziegler-Nichols PI - TuneRuleTIR = 5: Ziegler-Nichols P
PIDSelfTune.TIR.State	INT	0	Variable TIR.State indicates the current phase of "fine tuning":
Retain.Mode	INT	0	A change to the value of Retain.Mode initiates a change of the operating mode. The following operating mode is enabled upon a change from Mode to: - Mode = 0: inactive - Mode = 1: pretuning - Mode = 2: fine tuning - Mode = 3: automatic mode - Mode = 4: manual mode - Mode = 5: substitute output value approach - Mode = 6: transition time measurement - Mode = 7: substitute output value approach with error monitoring - Mode = 8: Error monitoring

Variable	Data type	Default	Description
Retain.CtrlParams.SetByUser	BOOL	FALSE	The PID parameters will be established automatically and PID_3Step will operate with a dead band at the output value if SetByUser = FALSE. The dead band width will be calculated during controller tuning on the basis of the standard deviation of the output value and saved in Retain.CtrlParams.OutputDeadBand. The PID parameters are entered manually and PID_3 Step operates without a dead band at the output value when SetByUser = TRUE. Retain.CtrlParams.OutputDeadBand = 0.0
Retain.CtrlParams.Gain	REAL	1.0	Active proportional gain
Retain.CtrlParams.Ti	REAL	20.0	• Ti > 0.0: active integral action time • Ti = 0.0: Integral action is disabled
Retain.CtrlParams.Td	REAL	0.0	• Td > 0.0: Active derivative action time • Td = 0.0: Derivative action is disabled
Retain.CtrlParams.TdFiltRatio	REAL	0.0	Active coefficient for derivative action delay
Retain.CtrlParams.PWeighting	REAL	0.0	Active proportional action weighting
Retain.CtrlParams.DWeighting	REAL	0.0	Active derivative action weighting
Retain.CtrlParams.Cycle	REAL	1.0	Active Sampling time PID algorithm, rounded to an integer multiple of the cycle time of the calling OB.
Retain.CtrlParams.InputDead Band	REAL	0.0	Dead band width of the control deviation

Note

Edit the variables listed in this table in "Inactive" mode to avoid malfunction of the PID controller. "Inactive" mode is forced by setting variable "Retain.Mode" to "0".

See also

State and Retain.Mode parameters (Page 1804)

ActivateRecoverMode variable (Page 1813)

State and Retain.Mode parameters

Correlation of the parameters

The State parameter indicates the current operating mode of the PID controller. You cannot modify the State parameter.

You need to modify the Retain.Mode tag to change the operating mode. This also applies when the value for the new operating mode is already in Retain.Mode. First set Retain.Mode = 0and thenRetain.Mode = 3. Provided the current operating mode of the controller supports this change, State will be set to the value of Retain.Mode.

When PID_3Step automatically switches the operating mode, the following applies:
State != Retain.Mode.

Examples:

- After successful pretuning
  State = 3 and Retain.Mode = 1
- Error
  State = 0 and Retain.Mode remains at the same value, e.g Retain.Mode = 3
- ManualEnalbe = TRUE
  State = 4 and Retain.Mode remain at the previous value, for example, Retain.Mode = 3

Note

You wish to repeat successful fine tuning without exiting automatic mode with Mode = 0.

Setting Retain.Mode to an invalid value such as 9999 for one cycle has no effect on State. Set Mode = 2 in the next cycle. You can generate a change to Retain.Mode without first switching to "inactive" mode.

Meaning of values

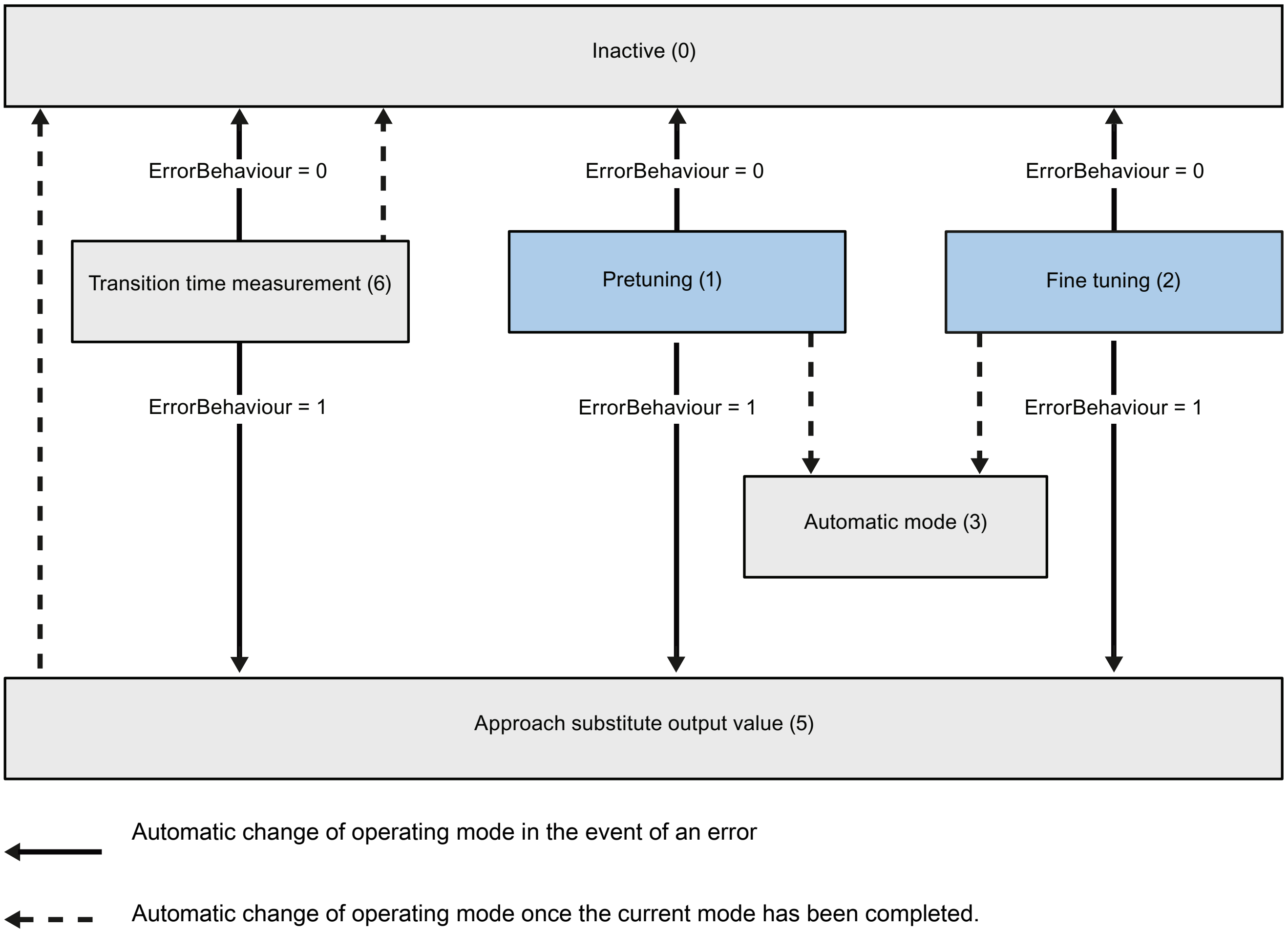
State / Retain.Mode	Description
0	Inactive The controller is switched off and no longer modifies the valve position.
1	Pretuning The pretuning determines the process response to a pulse of the output value and searches for the point of inflection. The optimized PID parameters are calculated as a function of the maximum rate of rise and dead time of the controlled system. Pretuning requirements: • State = 0 or State = 4 • ManualEnable = FALSE • The motor transition time has been configured or measured. • The setpoint and the process value lie within the configured limits. The higher the stability of the process value, the easier it is to calculate the PID parameters and increase precision of the result. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. PID parameters are backed up before they are recalculated and can be reactivated with Config.LoadBackUp. The setpoint is frozen in variable CurrentSetpoint. There is a change to automatic mode following successful pretuning and to "inactive" mode following unsuccessful pretuning. The pretuning phase is indicated by variable SUT.State.

State / Retain.Mode	Description
2	Fine tuning Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are optimized based on the amplitude and frequency of this oscillation. The differences between the process response during pretuning and fine tuning are analyzed. All PID parameters are recalculated on the basis of the findings. PID parameters from fine tuning usually have better master control and disturbance behavior than PID parameters from pretuning. PID_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before fine tuning. They can be recovered with Config.LoadBackUp. The setpoint is frozen in variable CurrentSetpoint. Requirements for fine tuning: • The motor transition time has been configured or measured. • The setpoint and the process value lie within the configured limits. • ManualEnable = FALSE • Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode Fine tuning proceeds as follows when started in: • Automatic mode (State = 3) Start fine tuning in automatic mode if you wish to improve the existing PID parameters using controller tuning. PID_3Step will regulate using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. • Inactive (State = 0) or manual (State = 4) mode Pretuning is always started first. The PID parameters established will be used for adjustment until the control loop has stabilized and the requirements for fine tuning have been met. If PIDSelfTune.TIR.RunIn = TRUE, pretuning is skipped and an attempt is made to reach the setpoint with minimum or maximum output value. This can produce increased overshoot. Fine tuning will then start automatically. The controller will change to automatic mode after successfully completed fine tuning and to "inactive" mode if fine tuning has not been successfully completed. The fine tuning phase is indicated by variable TIR.State.
3	Automatic mode In automatic mode, PID_3Step corrects the controlled system in accordance with the parameters specified. The controller changes to automatic mode if one the following requirements is fulfilled: • Pretuning successfully completed • Fine tuning successfully completed • Change of variable Retain.Mode to the value 3. After CPU startup or change from Stop to RUN mode, PID_3Step will start in the most recently active operating mode. To retain PID_3Step in "Inactive" mode, set RunModeByStartup = FALSE. The ActivateRecoverMode variable is considered in automatic mode.

State / Retain.Mode	Description
4	Manual mode In manual mode, you specify the manual output values at the parameters Manual_UP and Manual_DN or ManualValue . If you are able to move the actuator to the output value in the event of an error is described in the ErrorBits parameter. This operating mode is enabled if Retain.Mode = 4, or at the rising edge on ManualEnable. If ManualEnable changes to TRUE, only State will change. Retain.Mode will retain its current value. At a falling edge on ManualEnable, PID_3Step returns to the previous operating mode. The change to automatic mode is bumpless. **PID_3Step V1.1** Manual mode is always possible in the event of an error. **PID_3Step V1.0** Manual mode depends on the ActivateRecoverMode variable in the event of an error.
5	Approach substitute output value This operating mode is activated in the event of an error or when Reset = TRUE if Errorbehaviour = 1 and ActivateRecoverMode = FALSE.. PID_3Step moves the actuator to the substitute output value and then changes to "Inactive" mode.
6	Transition time measurement Determines the time that the motor needs to completely open the closed valve. This operating mode is activated when GetTransitTime.Start = TRUE is set. If endstop signals are used to measure transition time, the valve will be opened completely, closed completely and opened completely again starting at its current position. If GetTransitTime.InvertDirection = TRUE, this behavior is inverted. If a position feedback is used to measure transition time, the actuator will be moved from its current position to a target position.
7	Approach substitute output value with error monitoring This operating mode is activated instead of "Approach substitute output value" when an error occurs or with Reset = TRUE. PID_3Step moves the actuator to the substitute output value and then changes to "Error monitoring" mode. All the following conditions must be met: • Mode = 3 (automatic mode) • Errorbehaviour = 1 • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode (Page 1813) becomes effective. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.
8	Error monitoring The control algorithm is switched off and no longer changes the valve position. This operating mode is activated instead of "Inactive" mode in the event of an error or when Reset = TRUE. All the following conditions must be met: • Mode = 3 (automatic mode) • Errorbehaviour = 0 • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode (Page 1813) becomes effective. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.

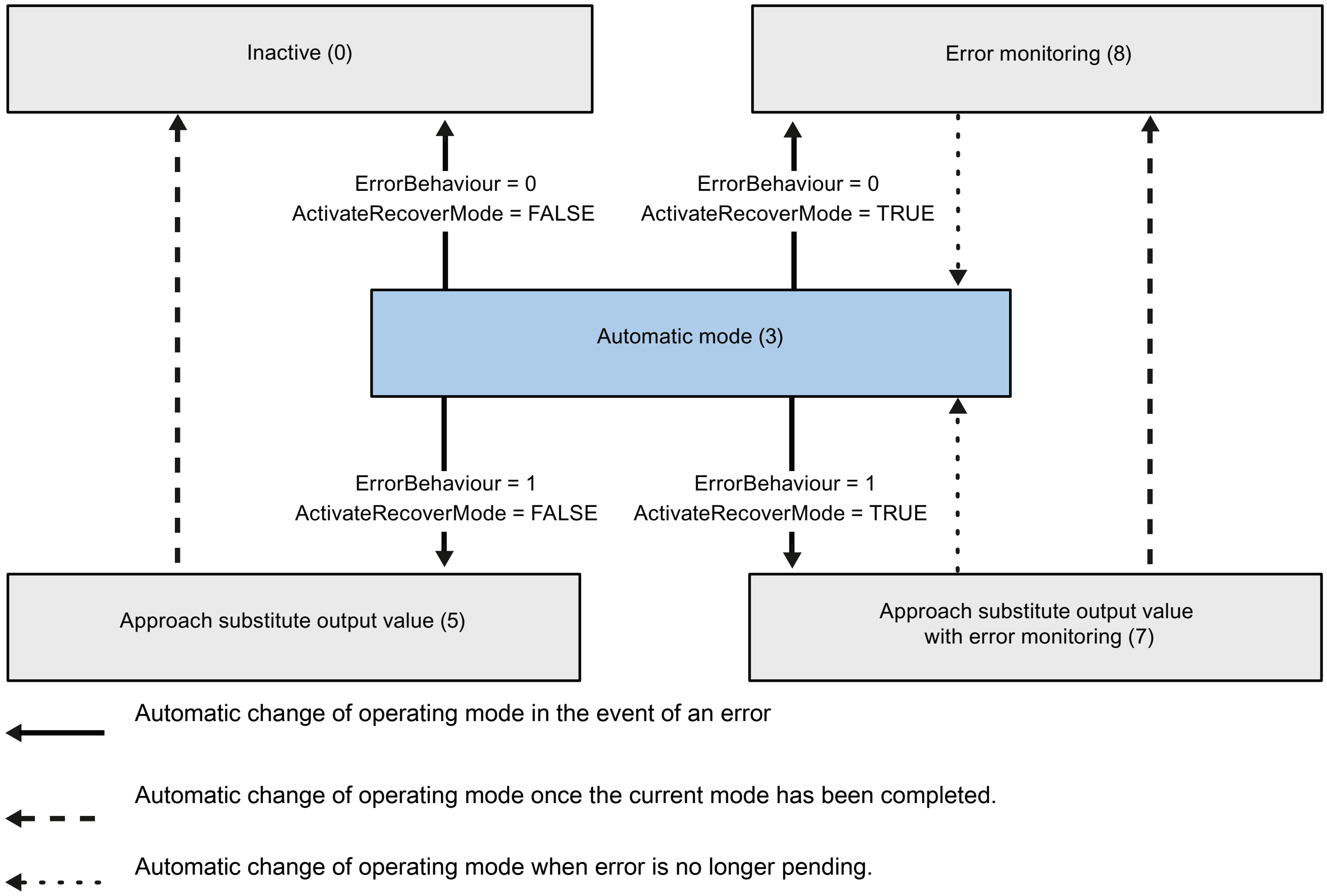
Automatic operating mode changes during commissioning

PID_3Step will automatically change operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour on the change of operating mode from transition time measurement, pretuning and fine tuning mode.


Inactive (0)
ErrorBehaviour = 0
ErrorBehaviour = 0
ErrorBehaviour = 0
Transition time measurement (6)
Pretuning (1)
Fine tuning (2)
ErrorBehaviour = 1
ErrorBehaviour = 1
ErrorBehaviour = 1
Automatic mode (3)
Approach substitute output value (5)
Automatic change of operating mode in the event of an error
Automatic change of operating mode once the current mode has been completed.

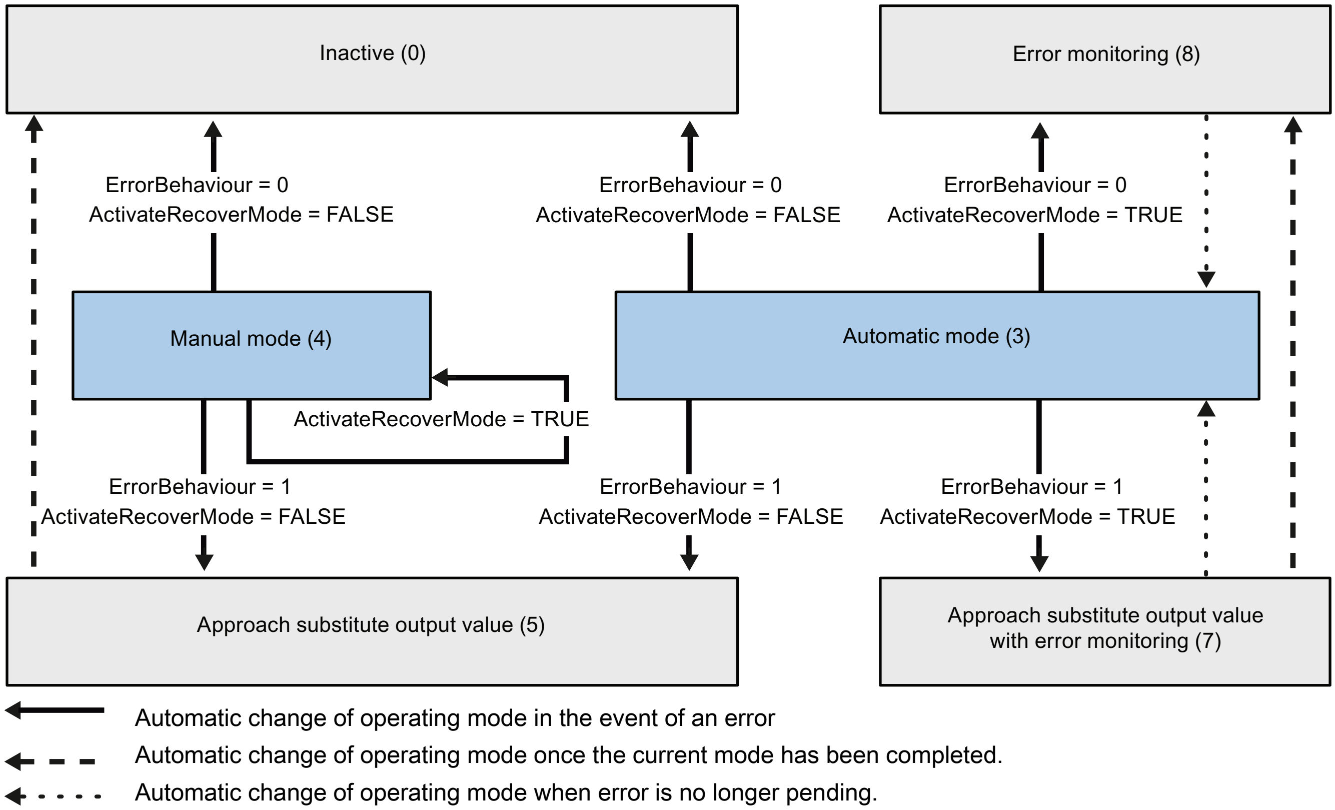
Automatic change of operating mode in automatic mode (PID_3Step V1.1)

PID_3Step will automatically change operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour and ActivateRecoverMode on the change of operating mode.


Inactive (0)
Error monitoring (8)
ErrorBehaviour = 0
ActivateRecoverMode = FALSE
ErrorBehaviour = 0
ActivateRecoverMode = TRUE
Automatic mode (3)
ErrorBehaviour = 1
ActivateRecoverMode = FALSE
ErrorBehaviour = 1
ActivateRecoverMode = TRUE
Approach substitute output value (5)
Approach substitute output value
with error monitoring (7)
Automatic change of operating mode in the event of an error
Automatic change of operating mode once the current mode has been completed.
Automatic change of operating mode when error is no longer pending.

Automatic operating mode changes during automatic and manual mode (PID_3Step V1.0)

PID_3Step will automatically change operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour and ActivateRecoverMode on the change of operating mode.


Inactive (0)
Error monitoring (8)
ErrorBehaviour = 0
ActivateRecoverMode = FALSE
ErrorBehaviour = 0
ActivateRecoverMode = FALSE
ErrorBehaviour = 0
ActivateRecoverMode = TRUE
Manual mode (4)
Automatic mode (3)
ActivateRecoverMode = TRUE
ErrorBehaviour = 1
ActivateRecoverMode = FALSE
ErrorBehaviour = 1
ActivateRecoverMode = FALSE
ErrorBehaviour = 1
ActivateRecoverMode = TRUE
Approach substitute output value (5)
Approach substitute output value with error monitoring (7)
Automatic change of operating mode in the event of an error
Automatic change of operating mode once the current mode has been completed.
Automatic change of operating mode when error is no longer pending.

See also

ActivateRecoverMode variable (Page 1813)

ErrorBits parameter (Page 1811)

ErrorBits parameter

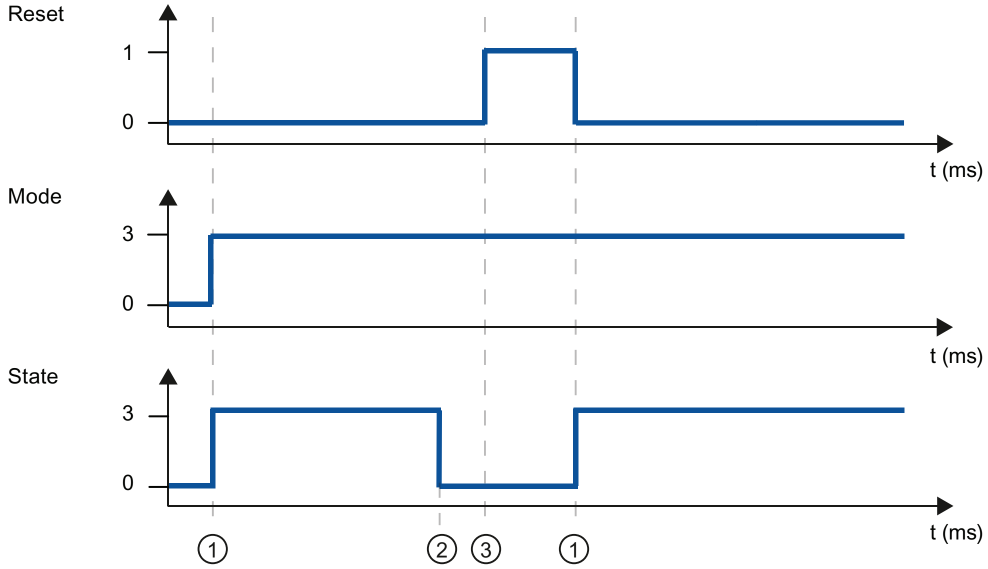
If several errors are pending, the values of the error codes are displayed by means of binary addition. The display of error code 0003, for example, indicates that the errors 0001 and 0002 are also pending.

ErrorBits (DW#16#...)	Description
0000	There is no error.
0001	The "Input" parameter is outside the process value limits. • Input > Config.InputUpperLimit or • Input < Config.InputLowerLimit If ActivateRecoverMode = TRUE and ErrorBehaviour = 1, the actuator moves to the substitute output value. If ActivateRecoverMode = TRUE and ErrorBehaviour = 0, the actuator stops in its current position. If ActivateRecoverMode = FALSE, the actuator stops in its current position. PID_3Step V1.1 You can move the actuator in manual mode. PID_3Step V1.0 Manual mode is not possible in this state. You cannot start the actuator again until you eliminate the error.
0002	Invalid value at parameter "Input_PER". Check whether an error is pending at the analog input. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE and the error is no longer pending, PID_3Step switches back to automatic mode.
0004	Error during fine tuning Oscillation of the process value could not be maintained.
0008	Error while starting pretuning. The process value is too close to the setpoint. Start fine tuning.
0010	The setpoint may not be changed during fine tuning.
0020	Pretuning may not be carried out in automatic mode or during fine tuning.
0040	Error in fine tuning The setpoint is too close to the setpoint limits.
0080	Error in pretuning. Incorrect configuration of output value limits. Check to see if the limits of the output value are configured correctly and match the direction in which the control is operating.
0100	Error during fine tuning has resulted in invalid parameters.
0200	Invalid value at parameter "Input": Numerical format of value is invalid. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE and the error is no longer pending, PID_3Step switches back to automatic mode.
0400	Calculating the output value failed. Check the PID parameters.
0800	Sampling time error: PID_3Step is not called within the sampling time of the cyclic interrupt OB. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE and the error is no longer pending, PID_3Step switches back to automatic mode.
1000	Invalid value at parameter "Setpoint": Numerical format of value is invalid. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE and the error is no longer pending, PID_3Step switches back to automatic mode.

ErrorBits (DW#16#...)	Description
2000	Invalid value at parameter Feedback_PER. Check whether an error is pending at the analog input. The actuator cannot be moved to the substitute output value and does not move from the current position. Manual mode is not possible in this state. You have to disable position feedback (Config. FeedbackOn = FALSE) to move the actuator from this state. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE and the error is no longer pending, PID_3Step switches back to automatic mode.
4000	Invalid value at parameter Feedback. Numerical format of value is invalid. The actuator cannot be moved to the substitute output value and does not move from the current position. Manual mode is not possible in this state. You have to disable position feedback (Config. FeedbackOn = FALSE) to move the actuator from this state. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE and the error is no longer pending, PID_3Step switches back to automatic mode.
8000	Error in digital position feedback. Actuator_H = TRUE and Actuator_L = TRUE. The actuator cannot be moved to the substitute output value and does not move from the current position. Manual mode is not possible in this state. You have to disable "Endstop signals actuator" (Config.ActuatorEndStopOn = FALSE) to move the actuator from this state. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE and the error is no longer pending, PID_3Step switches back to automatic mode.

Reset parameter

A rising edge at Reset resets the errors and warnings and clears the integral action. A falling edge at Reset triggers a change to the most recently active operating mode.



① Activation
② Error
③ Reset

ActivateRecoverMode variable

The effect of the ActivateRecoverMode variable depends on the version of the PID_3Step.

Behavior in version 1.1

The ActivateRecoverMode variable determines the behavior in the event of an error in automatic mode. ActivateRecoverMode is not effective during pretuning, fine tuning and transition time measurement.

ActivateRecover Mode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" operating mode. The controller is activated by a reset or a change in Retain.Mode.
TRUE	If errors occur frequently in automatic mode, this setting has a negative effect on the control response. In this case, check the ErrorBits parameter and eliminate the cause of the error. If one or more errors occur, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode: • 0002h: Invalid value at parameter Input_PER. • 0200h: Invalid value at parameter Input. • 0800h: Sampling time error • 1000h: Invalid value at parameter Setpoint. • 2000h: Invalid value at parameter Feedback_PER. • 4000h: Invalid value at parameter Feedback. • 8000h: Error in digital position feedback. With errors 2000h, 4000h and 8000h, PID_3Step cannot approach the configured substitute output value. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.

Behavior in version 1.0

The ActivateRecoverMode variable determines the behavior in the event of an error in automatic and manual mode. ActivateRecoverMode is not effective during pretuning, fine tuning and transition time measurement.

ActivateRecover Mode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" operating mode. The controller is activated by a reset or a change in Retain.Mode.
TRUE	**Errors in automatic mode** If errors occur frequently in automatic mode, this setting has a negative effect on the control response. In this case, check the ErrorBits parameter and eliminate the cause of the error. If one or more errors occur, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode: • 0002h: Invalid value at parameter Input_PER. • 0200h: Invalid value at parameter Input. • 0800h: Sampling time error • 1000h: Invalid value at parameter Setpoint. • 2000h: Invalid value at parameter Feedback_PER. • 4000h: Invalid value at parameter Feedback. • 8000h: Error in digital position feedback. With errors 2000h, 4000h and 8000h, PID_3Step **cannot** approach the configured substitute output value. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode. **Errors in manual mode** If one or more of the following errors occur, PID_3Step stays in manual mode: • 0002h: Invalid value at parameter Input_PER. • 0200h: Invalid value at parameter Input. • 0800h: Sampling time error • 1000h: Invalid value at parameter Setpoint. • 2000h: Invalid value at parameter Feedback_PER. • 4000h: Invalid value at parameter Feedback. • 8000h: Error in digital position feedback. With errors 2000h, 4000h and 8000h, you **cannot** move the valve to a suitable position.

See also

PID_3Step static variables (Page 1797)

State and Retain.Mode parameters (Page 1804)

Warnings variable

If several warnings are pending, their values are displayed by means of binary addition. The display of warning 0003, for example, indicates that the warnings 0001 and 0002 are also pending.

Warnings (DW#16#...)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0002	Oscillation increased during fine tuning.
0004	The setpoint was limited to the configured limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. The PID parameters were instead calculated using the TuneRuleTIR = 3 method.
0010	The operating mode could not be changed because ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.
0080	Invalid value at Retain.Mode. The operating mode is not changed.
0100	The manual value was limited to the limits of the controller output.
0200	The rule used for controller tuning returns incorrect results, or is not supported.
0400	Method selected for transition time measurement not suitable for actuator. The transition time cannot be measured because the actuator settings do not match the selected measuring method.
0800	The difference between the current position and the new output value is too small for transition time measurement. This state can produce incorrect results. The difference between the current output value and new output value must be at least 50% of the entire control range.
1000	The substitute output value cannot be reached because it is beyond the output value limits.

The following warnings are deleted as soon as the cause is dealt with:

- 0004
- 0020
- 0040
- 0100

All other warnings are cleared with a rising edge at Reset.

SUT.State variable

SUT.State	Name	Description
0	SUT_INIT	Initialize pretuning
50	SUT_TPDN	Determine start position without position feedback
100	SUT_STDABW	Calculate the standard deviation
200	SUT_GET_POI	Find the point of inflection
300	SUT_GET_RISETM	Determine the rise time
9900	SUT_IO	Pretuning successful
1	SUT_NIO	Pretuning not successful

TIR.State variable

TIR.State	Name	Description
-100	TIR_FIRST_SUT	Fine tuning is not possible. Pretuning will be executed first.
0	TIR_INIT	Initialize fine tuning
200	TIR_STDABW	Calculate the standard deviation
300	TIR_RUN_IN	Attempt to reach the setpoint with the maximum or minimum output value
400	TIR_CTRLN	Attempt to reach the setpoint with the existing PID parameters (if pretuning has been successful)
500	TIR_OSZIL	Determine oscillation and calculate parameters
9900	TIR_IO	Fine tuning successful
1	TIR_NIO	Fine tuning not successful

9.8.4.2 Motion Control

S7-1200 Motion Control

MC_Power

MC_Power: Enable, disable axis

Description

The Motion Control instruction "MC_Power" releases or locks an axis.

Requirements

- The technology object "Axis" has been configured correctly.
- There is no pending enable-inhibiting error.

Override response

Execution of "MC_Power" cannot be aborted by a motion control command.

Disabling the axis (input parameter "Enable" = FALSE) aborts all motion control jobs for the associated technology object in accordance with the selected "StopMode".

Parameter

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Enable	INPUT	BOOL	FALSE	TRUE	Motion Control attempts to enable the axis.
				FALSE	All active jobs are aborted according to the configured "StopMode" and the axis is stopped.
StopMode	INPUT	INT	0	0	Emergency stop If a request to disable the axis is pending, the axis brakes at the configured emergency stop deceleration. The axis is disabled after reaching standstill.
				1	Immediate stop If a request to disable the axis is pending, this axis is disabled without deceleration. Pulse output is stopped immediately.
Status	OUTPUT	BOOL	FALSE	Status of axis enable	
				FALSE	The axis is disabled. The axis does not execute motion control jobs and does not accept any new jobs (exception: MC_Reset command). The axis is not homed. Upon disabling, the status does not change to FALSE until the axis reaches a standstill.
				TRUE	The axis is enabled. The axis is ready to execute motion control jobs. Upon axis enabling, the status does not change to TRUE until the signal "Drive ready" is pending. If the "Drive ready" drive interface was not configured in the axis configuration, the status changes to TRUE immediately.
Busy	OUTPUT	BOOL	FALSE	TRUE	MC Power is active
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred in motion control instruction "MC_Power" or in the associated technology object. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".

Parameter	Declaration	Data type	Default value	Description
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 3028) for parameter "Error""
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 3028) for parameter "ErrorID"

NOTICE
If the axis is switched off due to an error, it will be enabled again automatically after the error has been eliminated and acknowledged. This requires that input parameter "Enable" has retained the value TRUE during this process.

Enabling an axis with configured drive interface

To enable the axis, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameter "StopMode" with the desired value. Set input parameter "Enable" to TRUE.

   The enable output for "Drive enabled" changes to TRUE to enable the power to the drive. The CPU waits for the "Drive ready" signal of the drive.

   When the "Drive ready" signal is available at the configured ready input of the CPU, the axis becomes enabled. Output parameter "Status" and technology object tag <Axis name>.StatusBits.Enable indicate the value TRUE.

Enabling an axis without configured drive interface

To enable the axis, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameter "StopMode" with the desired value. Set input parameter "Enable" to TRUE. The axis is enabled. Output parameter "Status" and technology object tag <Axis name>.StatusBits.Enable indicate the value TRUE.

Disabling an axis

To disable an axis, you can follow the steps described below:

1. Bring the axis to a standstill.

   You can identify when the axis is at a standstill in the tag of the technology object <Axis name>.StatusBits.StandStill.
2. Set input parameter "Enable" to FALSE after standstill is reached.
3. If output parameters "Busy" and "Status" and technology object tag <Axis name>.StatusBits.Enable indicate the value FALSE, disabling of the axis is complete.

See also

List of ErrorIDs and ErrorInfos (technology objects as of V2.0) (Page 3028)

MC_Power: Function chart (Page 1820)

MC_Reset: Acknowledge error (Page 1821)

MC_Home: Home axes, set home position (Page 1822)

MC_Halt: Halt axis (Page 1826)

MC_MoveAbsolute: Absolute positioning of axes (Page 1829)

MC_MoveRelative: Relative positioning of axes (Page 1832)

MC_MoveVelocity: Move axes at preset rotational speed (Page 1836)

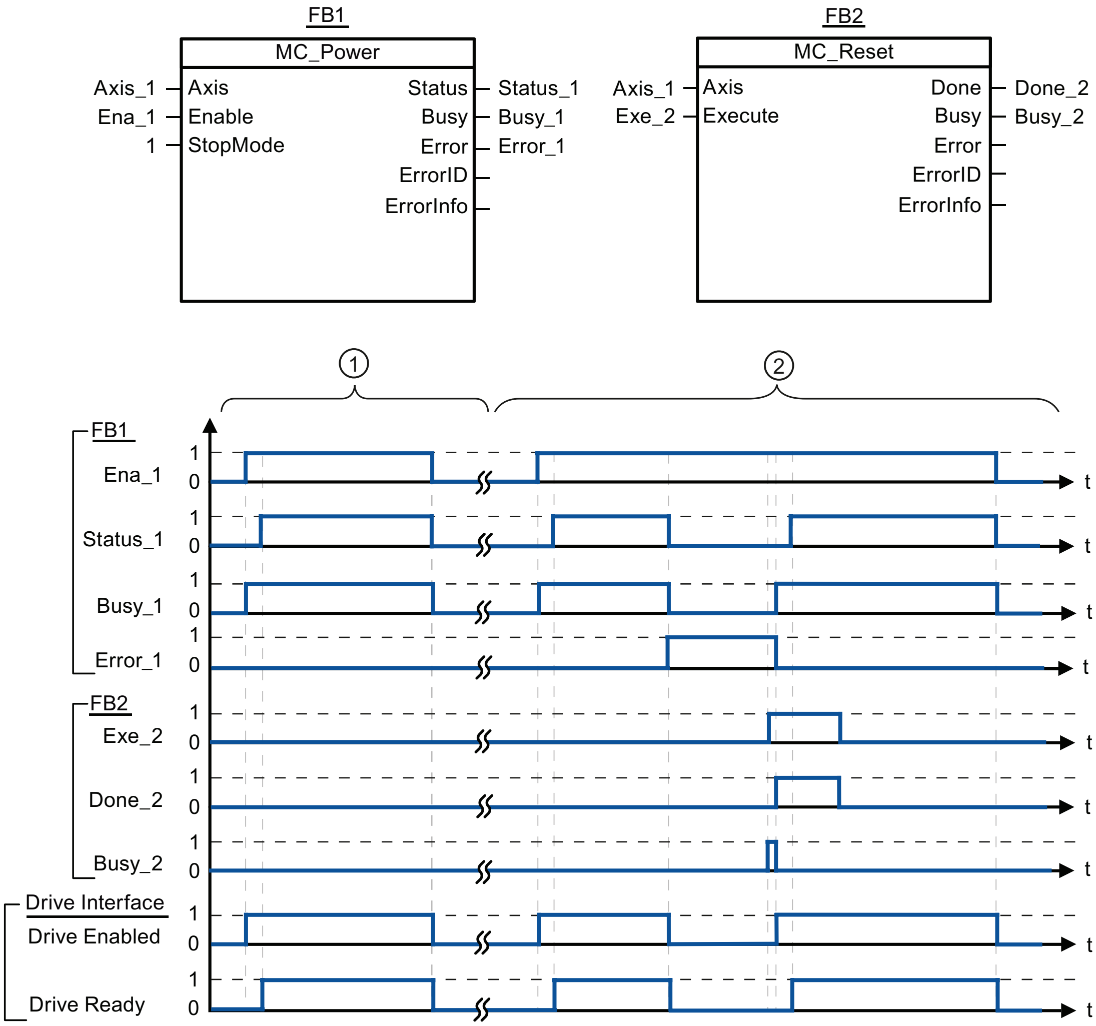
MC_MoveJog: Move axes in jogging mode (Page 1840)

MC_ChangeDynamic: Changing the dynamic settings for the axis (as of technology object "Axis" V2.0) (Page 1846)

MC_CommandTable: Run axis jobs as movement sequence (as of technology object "Axis" V2.0) (Page 1844)

MC_Power: Function chart

Function chart



①	An axis is enabled and then disabled again. When the drive has signaled "Drive ready" back to the CPU, the successful enable can be read out via "Status_1".
②	Following an axis enable, an error has occurred that caused the axis to be disabled. The error is eliminated and acknowledged with "MC_Reset". The axis is then enabled again.

See also

MC_Power: Enable, disable axis (Page 1816)

MC_Reset

MC_Reset: Acknowledge error

Description

Motion Control instruction "MC_Reset" can be used to acknowledge "Operating error with axis stop" and "Configuration error". The errors that require acknowledgement can be found in the "List of ErrorIDs and ErrorInfos" under "Remedy".

Requirements

- The technology object "Axis" has been configured correctly.
- The cause of a pending configuration error requiring acknowledgement has been eliminated (for example, acceleration in "Axis" technology object has been changed to a valid value).

Override response

The MC_Reset command cannot be aborted by any other motion control command.

The new MC_Reset command does not abort any other active motion control jobs.

Parameter

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Done	OUTPUT	BOOL	FALSE	TRUE	Error has been acknowledged.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 3028) for parameter "Error""	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 3028) for parameter "ErrorID"	

Acknowledging an error requiring acknowledgement with MC_Reset

To acknowledge an error, follow these steps:

1. Check the requirements indicated above.
2. Start the acknowledgement of the error with a rising edge at input parameter "Execute".
3. If output parameter "Done" indicates the value TRUE and technology object tag <Axis name>.StatusBits.Error the value FALSE, the error has been acknowledged.

See also

List of ErrorIDs and ErrorInfos (technology objects as of V2.0) (Page 3028)

MC_Power: Enable, disable axis (Page 1816)

MC_Home: Home axes, set home position (Page 1822)

MC_Halt: Halt axis (Page 1826)

MC_MoveAbsolute: Absolute positioning of axes (Page 1829)

MC_MoveRelative: Relative positioning of axes (Page 1832)

MC_MoveVelocity: Move axes at preset rotational speed (Page 1836)

MC_MoveJog: Move axes in jogging mode (Page 1840)

MC_CommandTable: Run axis jobs as movement sequence (as of technology object "Axis" V2.0) (Page 1844)

MC_ChangeDynamic: Changing the dynamic settings for the axis (as of technology object "Axis" V2.0) (Page 1846)

MC_Home

MC_Home: Home axes, set home position

Description

Use the motion control instruction "MC_Home" to match the axis coordinates to the real, physical drive position. Homing is required for absolute positioning of the axis. The following types of homing can be executed:

- Active homing (Mode = 3)

  The homing procedure is executed automatically.

- Passive homing (Mode = 2)

  During passive homing, the motion control instruction "MC_Home" does not carry out any homing motion. The traversing motion required for this step must be implemented by the user via other motion control instructions. When the homing switch is detected, the axis is homed.

- Direct homing absolute (Mode = 0)

  The current axis position is set to the value of parameter "Position".

- Direct homing relative (Mode = 1)

  The current axis position is offset by the value of parameter "Position".

Requirements

- The technology object "Axis" has been configured correctly.
- The axis is enabled.
- No MC_CommandTable command may be active upon start with Mode = 0, 1 or 2.

Override response

The override response is dependent on the selected mode:

Mode = 0, 1

The MC_Home command cannot be aborted by any other motion control command.

The MC_Home command does not abort any active motion control jobs. Position-related motion jobs are resumed after homing according to the new homing position (value at input parameter: "Position").

Mode = 2

The MC_Home command can be aborted by the following motion control jobs:

- MC_Home command Mode = 2, 3

The new MC_Home command aborts the following active motion control command.

- MC_Home command Mode = 2

Position-related motion jobs are resumed after homing according to the new homing position (value at input parameter: "Position").

Mode = 3

The MC_Home command can be aborted by the following motion control jobs:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command

The new MC_Home command aborts the following active motion control jobs:

- MC_Home command Mode = 2, 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command

Parameter

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Position	INPUT	REAL	0.0	• Mode = 0, 2, and 3 Absolute position of axis after completion of the homing operation • Mode = 1 Correction value for the current axis position Limit values: $-1.0e^{12} \leq \text{Position} \leq 1.0e^{12}$	
Mode	INPUT	INT	0	Homing mode	
				0	Direct homing absolute New axis position is the position value of parameter "Position".
				1	Direct homing relative New axis position is the current axis position + position value of parameter "Position".
				2	Passive homing Homing according to the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position.
				3	Active homing Reference point approach in accordance with the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position.
Done	OUTPUT	BOOL	FALSE	TRUE	Command completed
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID for parameter "Error""	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID for parameter "ErrorID"	

Note

Axis homing is lost under the following conditions:

- Disabling of axis by motion control instruction "MC_Power"
- Changeover between automatic mode and manual control
- Upon start of active homing. After successful completion of the homing operation, axis homing is again available.
- After POWER OFF -> POWER ON of the CPU
- After CPU restart (RUN-STOP -> STOP-RUN)

Homing an axis

To home the axis, follow these stops:

1. Check the requirements indicated above.
2. Initialize the necessary input parameters with values, and start the homing operation with a rising edge at input parameter "Execute"
3. If output parameter "Done" and technology object tag <Axis name>.StatusBits.HomingDone indicate the value TRUE, homing is complete.

See also

List of ErrorIDs and ErrorInfos (technology objects as of V2.0) (Page 3028)

MC_Power: Enable, disable axis (Page 1816)

MC_Reset: Acknowledge error (Page 1821)

MC_Halt: Halt axis (Page 1826)

MC_MoveAbsolute: Absolute positioning of axes (Page 1829)

MC_MoveRelative: Relative positioning of axes (Page 1832)

MC_MoveVelocity: Move axes at preset rotational speed (Page 1836)

MC_MoveJog: Move axes in jogging mode (Page 1840)

MC_CommandTable: Run axis jobs as movement sequence (as of technology object "Axis" V2.0) (Page 1844)

MC_ChangeDynamic: Changing the dynamic settings for the axis (as of technology object "Axis" V2.0) (Page 1846)

MC_Halt

MC_Halt: Halt axis

Description

The "MC_Halt" motion control instruction stops all movements and brings the axis to a standstill with the configured deceleration. The standstill position is not defined.

Requirements

- The technology object "Axis" has been configured correctly.
- The axis is enabled.

Override response

The MC_Halt command can be aborted by the following motion control jobs:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command

The new MC_Halt command aborts the following active motion control jobs:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command

Parameter

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Done	OUTPUT	BOOL	FALSE	TRUE	Zero velocity reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 3028) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 3028) for parameter "ErrorID"	

See also

MC_Halt: Function chart (Page 1828)

List of ErrorIDs and ErrorInfos (technology objects as of V2.0) (Page 3028)

MC_Power: Enable, disable axis (Page 1816)

MC_Reset: Acknowledge error (Page 1821)

MC_Home: Home axes, set home position (Page 1822)

MC_MoveAbsolute: Absolute positioning of axes (Page 1829)

MC_MoveRelative: Relative positioning of axes (Page 1832)

MC_MoveVelocity: Move axes at preset rotational speed (Page 1836)

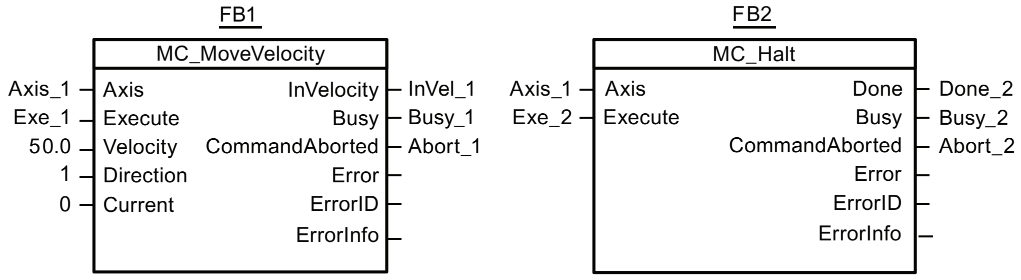
MC_MoveJog: Move axes in jogging mode (Page 1840)

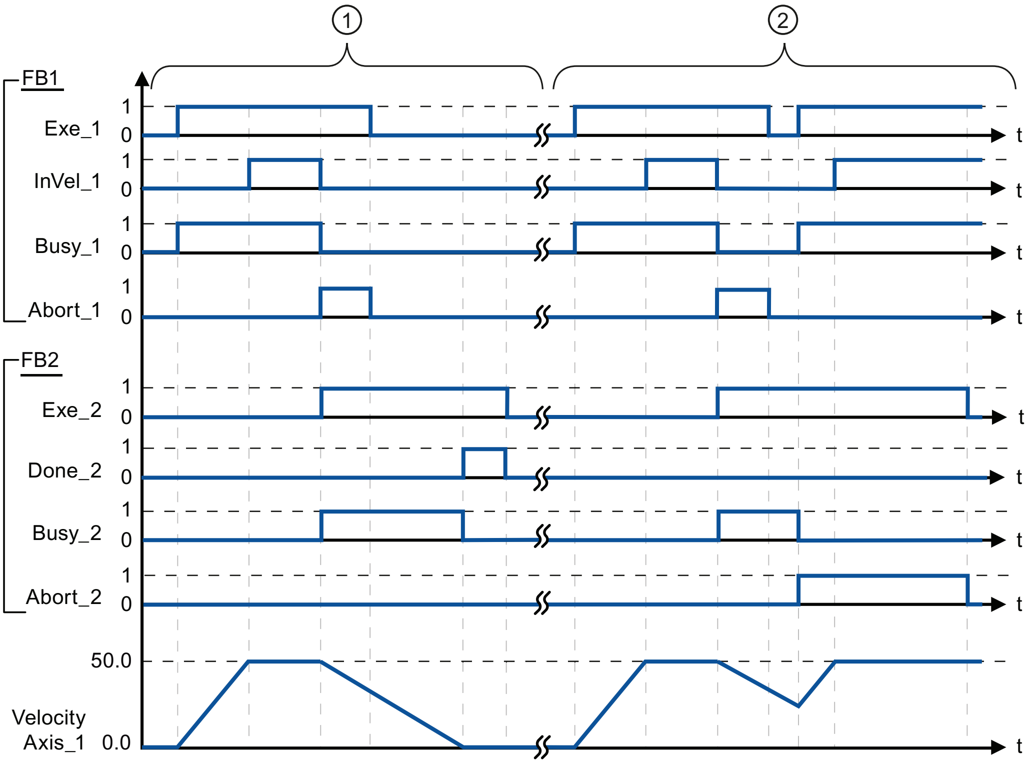
MC_CommandTable: Run axis jobs as movement sequence (as of technology object "Axis" V2.0) (Page 1844)

MC_ChangeDynamic: Changing the dynamic settings for the axis (as of technology object "Axis" V2.0) (Page 1846)

MC_Halt: Function chart

Function chart





The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 5.0

①	The axis is braked by an MC_Halt job until it comes to a standstill. The axis standstill is signaled via "Done_2".
②	While an MC_Halt job is braking the axis, this job is aborted by another motion job. The abort is signaled via "Abort_2".

See also

MC_Halt: Halt axis (Page 1826)

MC_MoveAbsolute

MC_MoveAbsolute: Absolute positioning of axes

Description

The "MC_MoveAbsolute" motion control instruction starts an axis positioning motion to move it to an absolute position.

Requirements

- The technology object "Axis" has been configured correctly.
- The axis is enabled.
- The axis is homed.

Override response

The MC_MoveAbsolute command can be aborted by the following motion control jobs:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command

The new MC_MoveAbsolute command aborts the following active motion control jobs:

- MC_Home command Mode = 3
- MC_Halt-Auftrag
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command

Parameter

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Position	INPUT	REAL	0.0	Absolute target position Limit values: $-1.0e^{12}$ ≤ Position ≤ $1.0e^{12}$	
Velocity	INPUT	REAL	10.0	Velocity of axis This velocity is not always reached on account of the configured acceleration and deceleration and the target position to be approached. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity	
Done	OUTPUT	BOOL	FALSE	TRUE	Absolute target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 3028) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 3028) for parameter "ErrorID"	

See also

MC_MoveAbsolute: Function chart (Page 1831)

List of ErrorIDs and ErrorInfos (technology objects as of V2.0) (Page 3028)

MC_Power: Enable, disable axis (Page 1816)

MC_Reset: Acknowledge error (Page 1821)

MC_Home: Home axes, set home position (Page 1822)

MC_Halt: Halt axis (Page 1826)

MC_MoveRelative: Relative positioning of axes (Page 1832)

MC_MoveVelocity: Move axes at preset rotational speed (Page 1836)

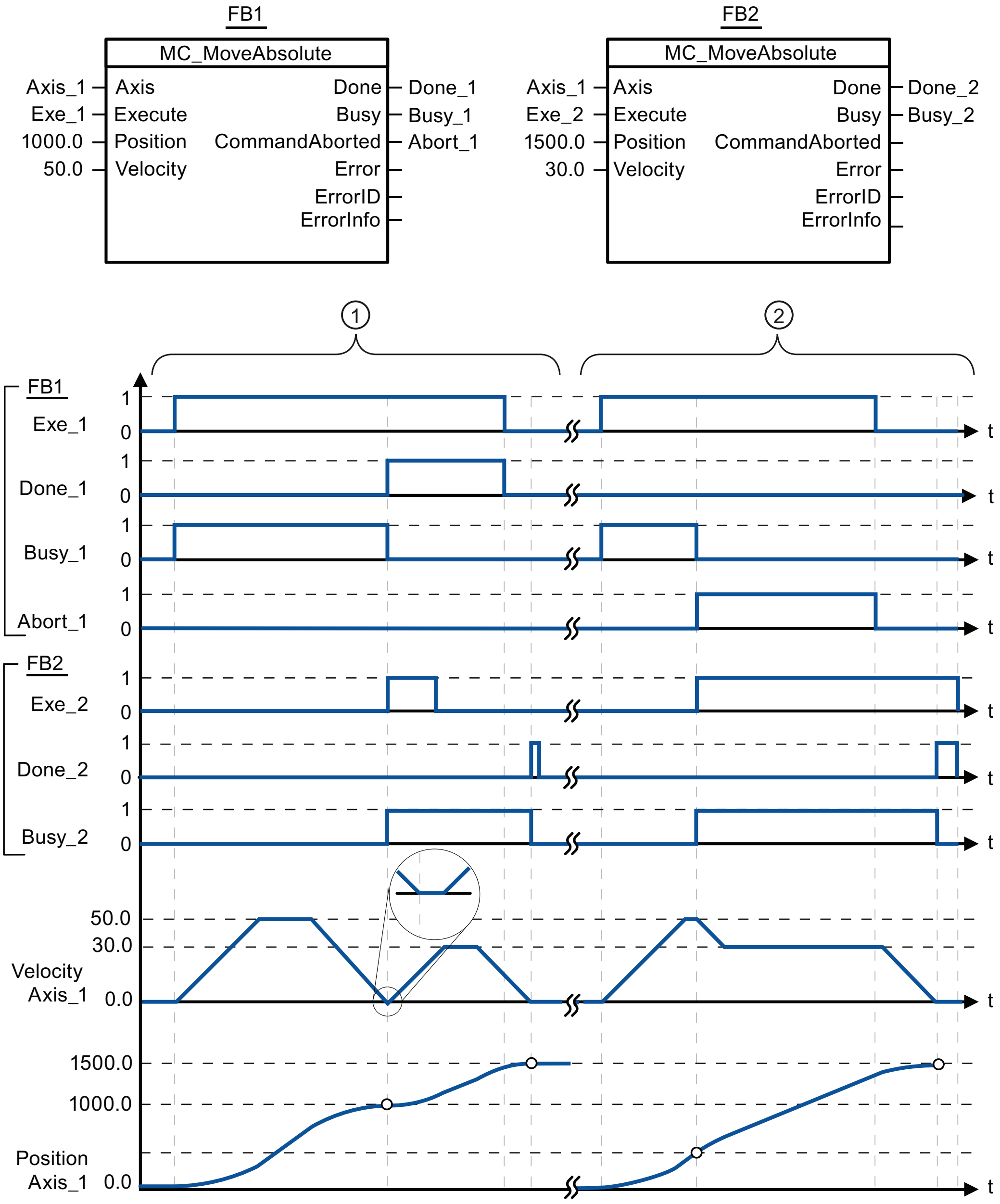
MC_MoveJog: Move axes in jogging mode (Page 1840)

MC_CommandTable: Run axis jobs as movement sequence (as of technology object "Axis" V2.0) (Page 1844)

MC_ChangeDynamic: Changing the dynamic settings for the axis (as of technology object "Axis" V2.0) (Page 1846)

MC_MoveAbsolute: Function chart

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	An axis is moved to absolute position 1000.0 with a MC_MoveAbsolute job. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveAbsolute job, with target position 1500.0, is started. Because of the response times (e.g., cycle time of user program, etc.), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".
②	An active MC_MoveAbsolute job is aborted by another MC_MoveAbsolute job. The abort is signaled via "Abort_1". The axis is then moved at the new velocity to the new target position 1500.0. When the new target position is reached, this is signaled via "Done_2".

See also

MC_MoveAbsolute: Absolute positioning of axes (Page 1829)

MC_MoveRelative

MC_MoveRelative: Relative positioning of axes

Description

The "MC_MoveRelative" motion control instruction starts a positioning motion relative to the start position.

Requirements

- The technology object "Axis" has been configured correctly.
- The axis is enabled.

Override response

The MC_MoveRelative command can be aborted by the following motion control jobs:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command

The new MC_MoveRelative command aborts the following active motion control jobs:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command

Parameter

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Distance	INPUT	REAL	0.0	Travel distance for the positioning operation Limit values: $-1.0e^{12} \le$ Distance $\le 1.0e^{12}$	
Velocity	INPUT	REAL	10.0	Velocity of axis This velocity is not always reached on account of the configured acceleration and deceleration and the distance to be traveled. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity	
Done	OUTPUT	BOOL	FALSE	TRUE	Target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 3028) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 3028) for parameter "ErrorID"	

See also

MC_MoveRelative: Function chart (Page 1835)

List of ErrorIDs and ErrorInfos (technology objects as of V2.0) (Page 3028)

MC_Power: Enable, disable axis (Page 1816)

MC_Reset: Acknowledge error (Page 1821)

MC_Home: Home axes, set home position (Page 1822)

MC_Halt: Halt axis (Page 1826)

MC_MoveAbsolute: Absolute positioning of axes (Page 1829)

MC_MoveVelocity: Move axes at preset rotational speed (Page 1836)

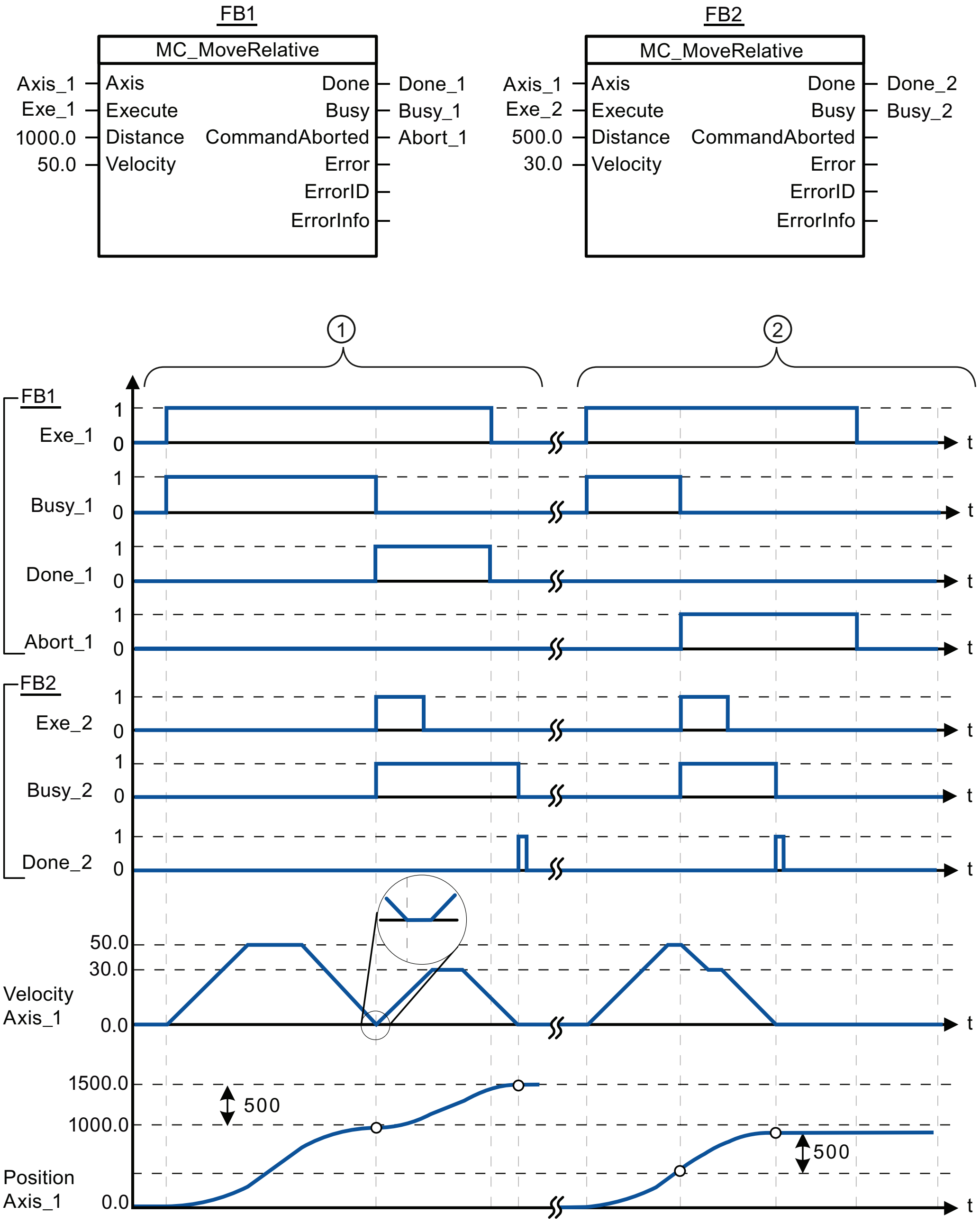
MC_MoveJog: Move axes in jogging mode (Page 1840)

MC_CommandTable: Run axis jobs as movement sequence (as of technology object "Axis" V2.0) (Page 1844)

MC_ChangeDynamic: Changing the dynamic settings for the axis (as of technology object "Axis" V2.0) (Page 1846)

MC_MoveRelative: Function chart

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	The axis is moved by an MC_MoveRelative job by the distance ("Distance") 1000.0. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveRelative job, with travel distance 500.0, is started. Because of the response times (e.g., cycle time of user program, etc.), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".
②	An active MC_MoveRelative job is aborted by another MC_MoveRelative job. The abort is signaled via "Abort_1". The axis is then moved at the new velocity by the new distance ("Distance") 500.0. When the new target position is reached, this is signaled via "Done_2".

See also

MC_MoveRelative: Relative positioning of axes (Page 1832)

MC_MoveVelocity

MC_MoveVelocity: Move axes at preset rotational speed

Description

Motion control instruction "MC_MoveVelocity" moves the axis constantly at the specified velocity.

Requirements

- The technology object "Axis" has been configured correctly.
- The axis is enabled.

Override response

MC_MoveVelocity can be aborted by the following motion control jobs:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command

The new MC_MoveVelocity command aborts the following active motion control jobs:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command

Parameter

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Velocity	INPUT	REAL	10.0	Velocity specification for axis motion Limit values: Start/stop velocity ≤ |Velocity| ≤ maximum velocity (Velocity = 0.0 is permitted)	
Direction	INPUT	INT	0	Direction specification	
				0	Direction of rotation corresponds to the sign of the value in parameter "Velocity"
				1	Positive direction of rotation (The sign of the value in parameter "Velocity" is ignored)
				2	Negative direction of rotation (The sign of the value in parameter "Velocity" is ignored)
Current	INPUT	BOOL	FALSE	Maintain current velocity	
				FALSE	"Maintain current velocity" is deactivated. The values of parameters "Velocity" and "Direction" are used.
				TRUE	"Maintain current velocity" is activated. The values in parameters "Velocity" and "Direction" are not taken into account. When the axis resumes motion at the current velocity, the "InVelocity" parameter returns the value TRUE.
InVelocity	OUTPUT	BOOL	FALSE	TRUE	• "Current" = FALSE: The velocity specified in parameter "Velocity" was reached. • "Current" = TRUE: The axis travels at the current velocity at the start time.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.

Parameter	Declaration	Data type	Default value	Description	
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 3028) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 3028) for parameter "ErrorID"	

Behavior with zero set velocity (Velocity = 0.0)

An MC_MoveVelocity command with "Velocity" = 0.0 (such as an MC_Halt command) aborts active motion jobs and stops the axis with the configured deceleration.

When the axis comes to a standstill, output parameter "InVelocity" indicates TRUE for at least one program cycle.

"Busy" indicates the value TRUE during the deceleration operation and changes to FALSE together with "InVelocity". If parameter "Execute" = TRUE is set, "InVelocity" and "Busy" are latched.

When the MC_MoveVelocity command is started, status bit "SpeedCommand" is set in the technology object. Status bit "ConstantVelocity" is set upon axis standstill. Both bits are adapted to the new situation when a new motion command is started.

See also

MC_MoveVelocity: Function chart (Page 1839)

List of ErrorIDs and ErrorInfos (technology objects as of V2.0) (Page 3028)

MC_Power: Enable, disable axis (Page 1816)

MC_Reset: Acknowledge error (Page 1821)

MC_Home: Home axes, set home position (Page 1822)

MC_Halt: Halt axis (Page 1826)

MC_MoveAbsolute: Absolute positioning of axes (Page 1829)

MC_MoveRelative: Relative positioning of axes (Page 1832)

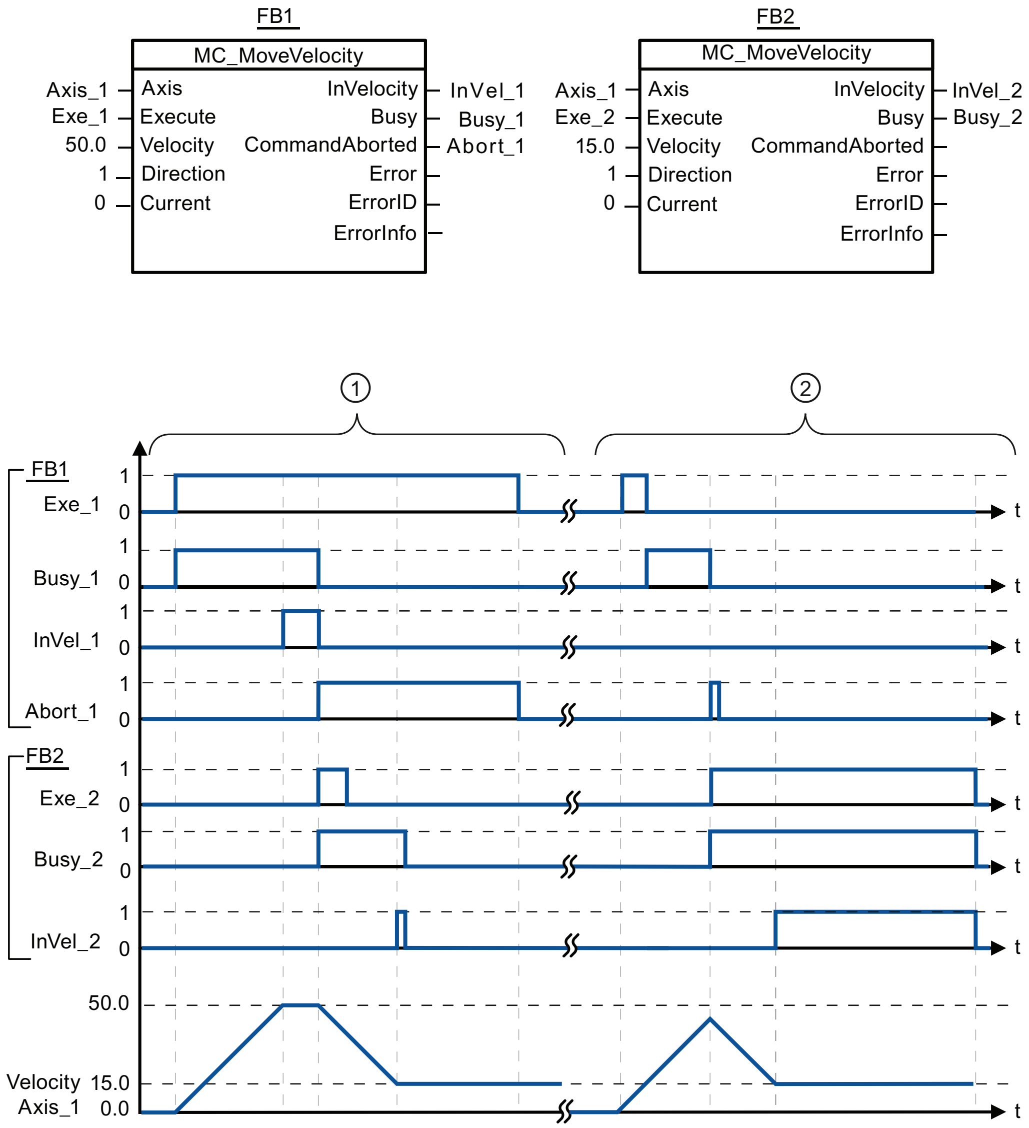
MC_MoveJog: Move axes in jogging mode (Page 1840)

MC_CommandTable: Run axis jobs as movement sequence (as of technology object "Axis" V2.0) (Page 1844)

MC_ChangeDynamic: Changing the dynamic settings for the axis (as of technology object "Axis" V2.0) (Page 1846)

MC_MoveVelocity: Function chart

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	An active MC_MoveVelocity job signals via "InVel_1" that its target velocity has been reached. It is then aborted by another MC_MoveVelocity job. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.
②	An active MC_MoveVelocity job is aborted by another MC_MoveVelocity job prior to reaching its target velocity. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.

See also

MC_MoveVelocity: Move axes at preset rotational speed (Page 1836)

MC_MoveJog

MC_MoveJog: Move axes in jogging mode

Description

Motion control instruction "MC_MoveJog" moves the axis constantly at the specified velocity in jog mode. You use this motion control instruction, for example, for testing and commissioning purposes.

Requirements

- The technology object "Axis" has been configured correctly.
- The axis is enabled.

Override response

The MC_MoveJog command can be aborted by the following motion control jobs:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command

The new MC_MoveJog command aborts the following active motion control jobs:

- MC_Home command (Mode = 3)
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command

Parameter

<table>
<tr><th>Parameter</th><th>Declaration</th><th>Data type</th><th>Default value</th><th colspan="2">Description</th></tr>
<tr><td>Axis</td><td>INPUT</td><td>TO_Axis_1</td><td>-</td><td colspan="2">Axis technology object</td></tr>
<tr><td>JogForward</td><td>INPUT</td><td>BOOL</td><td>FALSE</td><td colspan="2">As long as the parameter is TRUE, the axis moves in the positive direction at the velocity specified in parameter "Velocity".</td></tr>
<tr><td>JogBackward</td><td>INPUT</td><td>BOOL</td><td>FALSE</td><td colspan="2">As long as the parameter is TRUE, the axis moves in the negative direction at the velocity specified in parameter "Velocity".</td></tr>
<tr><td colspan="6">If both parameters are simultaneously TRUE, the axis stops with the configured deceleration. An error is indicated in parameters "Error", "ErrorID", and "ErrorInfo".</td></tr>
<tr><td rowspan="3">Velocity</td><td rowspan="3">INPUT</td><td rowspan="3">REAL</td><td rowspan="3">10.0</td><td colspan="2">Preset velocity for jog mode</td></tr>
<tr><td colspan="2">Limit values, instruction version V1.0:
Start/stop velocity ≤ |Velocity| ≤ maximum velocity</td></tr>
<tr><td colspan="2">Limits, instruction version V2.0:
Start/stop velocity ≤ velocity ≤ maximum velocity</td></tr>
<tr><td>InVelocity</td><td>OUTPUT</td><td>BOOL</td><td>FALSE</td><td>TRUE</td><td>The velocity specified in parameter "Velocity" was reached.</td></tr>
<tr><td>Busy</td><td>OUTPUT</td><td>BOOL</td><td>FALSE</td><td>TRUE</td><td>The command is being executed.</td></tr>
<tr><td>CommandAborted</td><td>OUTPUT</td><td>BOOL</td><td>FALSE</td><td>TRUE</td><td>During execution the command was aborted by another command.</td></tr>
<tr><td>Error</td><td>OUTPUT</td><td>BOOL</td><td>FALSE</td><td>TRUE</td><td>An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".</td></tr>
<tr><td>ErrorID</td><td>OUTPUT</td><td>WORD</td><td>16#0000</td><td colspan="2">Error ID (Page 3028) for parameter "Error"</td></tr>
<tr><td>ErrorInfo</td><td>OUTPUT</td><td>WORD</td><td>16#0000</td><td colspan="2">Error info ID (Page 3028) for parameter "ErrorID"</td></tr>
</table>

See also

MC_MoveJog: Function chart (Page 1843)

List of ErrorIDs and ErrorInfos (technology objects as of V2.0) (Page 3028)

MC_Power: Enable, disable axis (Page 1816)

MC_Reset: Acknowledge error (Page 1821)

MC_Home: Home axes, set home position (Page 1822)

MC_Halt: Halt axis (Page 1826)

MC_MoveAbsolute: Absolute positioning of axes (Page 1829)

MC_MoveRelative: Relative positioning of axes (Page 1832)

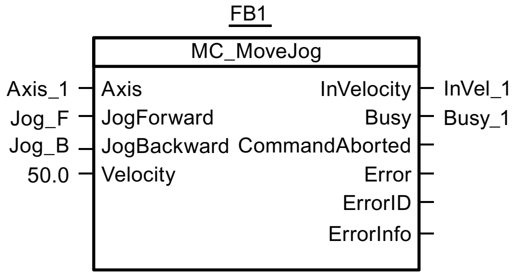
MC_MoveVelocity: Move axes at preset rotational speed (Page 1836)

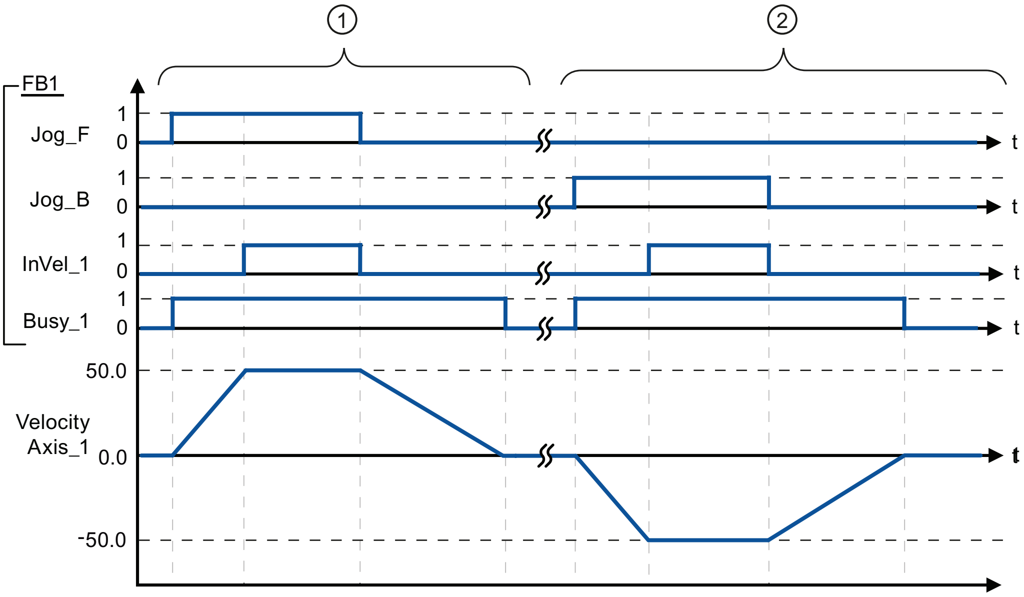
MC_CommandTable: Run axis jobs as movement sequence (as of technology object "Axis" V2.0) (Page 1844)

MC_ChangeDynamic: Changing the dynamic settings for the axis (as of technology object "Axis" V2.0) (Page 1846)

MC_MoveJog: Function chart

Function chart





The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 5.0

①	The axis is moved in the positive direction in jog mode via "Jog_F". When the target velocity 50.0 is reached, this is signaled via "InVelo_1". The axis brakes to a standstill again after Jog_F is reset.
②	The axis is moved in the negative direction in jog mode via "Jog_B". When the target velocity 50.0 is reached, this is signaled via "InVelo_1". The axis brakes to a standstill again after Jog_B is reset.

See also

MC_MoveJog: Move axes in jogging mode (Page 1840)

MC_CommandTable

MC_CommandTable: Run axis jobs as movement sequence (as of technology object "Axis" V2.0)

Description

The Motion Control instruction "MC_CommandTable" combines multiple individual axis control jobs in one movement sequence.

Requirements

- The technology object "Axis" has been added to Version V2.0 and correctly configured.
- The technology object "Command table" has been added and correctly configured.
- The axis is enabled.

Override response

The MC_CommandTable command can be aborted by the following motion control jobs:

- MC_Home command (Mode = 3)
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_CommandTable command aborts the following active motion control jobs:

- MC_Home command (Mode = 3)
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The active motion control command is cancelled by the start of the first "Positioning Relative", "Positioning Absolute", "Velocity set point" or "Halt" command.

Parameter

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
CommandTable	INPUT	TO_CommandTable_1	-	Command table technology object	
Execute	INPUT	BOOL	FALSE	Command table start with positive edge	
StartStep	INPUT	INT	1	Defines the step at which the execution of the command table should begin Limit values: 1 ≤ StartStep ≤ EndStep	
EndStep	INPUT	INT	32	Defines the step up to which the execution of command table should take place Limit values: StartStep ≤ EndStep ≤ 32	
Done	OUTPUT	BOOL	FALSE	TRUE	Command table has been successfully executed
Busy	OUTPU	BOOL	FALSE	TRUE	The command table is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The command table was cancelled by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command table. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 3028) for parameter "Error""	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 3028) for parameter "ErrorID"	
CurrentStep	OUTPUT	INT	0	Step in command table currently being executed	
StepCode	OUTPUT	WORD	16#0000	User-defined numerical value / bit pattern of the step currently being executed	

See also

List of ErrorIDs and ErrorInfos (technology objects as of V2.0) (Page 3028)

Overview of the Motion Control statements (Page 2991)

MC_Power: Enable, disable axis (Page 1816)

MC_Reset: Acknowledge error (Page 1821)

MC_Home: Home axes, set home position (Page 1822)

MC_Halt: Halt axis (Page 1826)

MC_MoveAbsolute: Absolute positioning of axes (Page 1829)

MC_MoveRelative: Relative positioning of axes (Page 1832)

MC_MoveVelocity: Move axes at preset rotational speed (Page 1836)

MC_MoveJog: Move axes in jogging mode (Page 1840)

MC_ChangeDynamic: Changing the dynamic settings for the axis (as of technology object "Axis" V2.0) (Page 1846)

MC_ChangeDynamic

MC_ChangeDynamic: Changing the dynamic settings for the axis (as of technology object "Axis" V2.0)

Description

Motion Control instruction "MC_ChangeDynamic" allows you to change the following settings of the axis:

- Change the ramp-up time (acceleration) value
- Change the ramp-down time (deceleration) value
- Change the emergency stop ramp-down time (emergency stop deceleration) value
- Change the smoothing time (jerk) value

The effectiveness of the change is shown in the description of the tag.

Requirements

- The technology object "Axis" has been added to Version V2.0.
- The technology object "Axis" has been configured correctly.

Override response

A MC_ChangeDynamic command cannot be aborted by any other Motion Control command.

A new MC_ChangeDynamic command does not abort any active Motion Control jobs.

Parameter

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
ChangeRampUp	INPUT	BOOL	FALSE	TRUE	Change ramp-up time in line with input parameter "RampUpTime"
RampUpTime	INPUT	REAL	5.00	Time (in seconds) to accelerate axis from standstill to configured maximum velocity without jerk limit. The change will influence the tag <Axis name>. Config.DynamicDefaults.Acceleration. The effectiveness of the change is shown in the description of this tag.	
ChangeRampDown	INPUT	BOOL	FALSE	TRUE	Change ramp-down time in line with input parameter "RampDownTime"

Parameter	Declaration	Data type	Default value	Description	
RampDown Time	INPUT	REAL	5.00	Time (in seconds) to decelerate axis from the configured maximum velocity to standstill without jerk limiter. The change will influence the tag <Axis name>. Config.DynamicDefaults.Deceleration . The effectiveness of the change is shown in the description of this tag.	
ChangeEm ergency	INPUT	BOOL	FALSE	TRUE	Change emergency stop ramp-down time in line with input parameter "EmergencyRampTime"
Emergency RampTime	INPUT	REAL	2.00	Time (in seconds) to decelerate the axis from configured maximum velocity to standstill without jerk limiter in emergency stop mode. The change will influence the tag <Axis name>. Config.DynamicDefaults.EmergencyDeceleration . The effectiveness of the change is shown in the description of this tag.	
ChangeJerk Time	INPUT	BOOL	FALSE	TRUE	Change smoothing time according to the input parameter "JerkTime"
JerkTime	INPUT	REAL	0.25	Smoothing time (in seconds) used for the axis acceleration and deceleration ramps The change will influence the tag <Axis name>. Config.DynamicDefaults.Jerk . The effectiveness of the change is shown in the description of this tag.	
Done	OUTPUT	BOOL	FALSE	TRUE	The changed values have been written to the technology data block. The description of the tags will show when the change becomes effective.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 3028) for parameter "Error""	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 3028) for parameter "ErrorID"	

Note

Values can be entered at the input parameters "RampUpTime", "RampDownTime", "EmergencyRampTime" and "JerkTime" which exceed the admissible limits of the resulting parameters: "Acceleration", "Deceleration", "Emergency stop deceleration" and "Jerk".

Please note the equations and limit values in "Axis technology object" -> "Configuring the technology object" -> "Dynamics" and ensure that the values you input are within the valid range.

See also

List of ErrorIDs and ErrorInfos (technology objects as of V2.0) (Page 3028)

Overview of the Motion Control statements (Page 2991)

Changing the configuration of dynamics in the user program (Page 2961)

Changing the homing configuration in the user program (Page 2968)

MC_Power: Enable, disable axis (Page 1816)

MC_Reset: Acknowledge error (Page 1821)

MC_Home: Home axes, set home position (Page 1822)

MC_Halt: Halt axis (Page 1826)

MC_MoveAbsolute: Absolute positioning of axes (Page 1829)

MC_MoveRelative: Relative positioning of axes (Page 1832)

MC_MoveVelocity: Move axes at preset rotational speed (Page 1836)

MC_MoveJog: Move axes in jogging mode (Page 1840)

MC_CommandTable: Run axis jobs as movement sequence (as of technology object "Axis" V2.0) (Page 1844)

9.8.4.3 High-speed counters

CTRL_HSC: Control high-speed counters

Parameter

Parameter	Data type	Memory area	Description
EN	BOOL	I, Q, M, D, L	Enable input
ENO	BOOL	I, Q, M, D, L	Enable output
HSC	HW_HSC	I, Q, M or constant	Hardware address of the high-speed counter (HW-ID)
DIR	BOOL	I, Q, M, D, L or constant	Enables the new count direction (see NEW_DIR)
CV	BOOL	I, Q, M, D, L or constant	Enables the new count value (see NEW_CV)
RV	BOOL	I, Q, M, D, L or constant	Enables the new reference value (see NEW_RV)
PERIOD	BOOL	I, Q, M, D, L or constant	Enables the new period of a frequency measurement (see NEW_PERIOD)

Parameter	Data type	Memory area	Description
NEW_DIR	INT	I, Q, M, D, L or constant	Count direction loaded when DIR = TRUE.
NEW_CV	DINT	I, Q, M, D, L or constant	Count value loaded when CV = TRUE.
NEW_RV	DINT	I, Q, M, D, L or constant	Reference value loaded when RV = TRUE.
NEW_PERIOD	INT	I, Q, M, D, L or constant	Period of the frequency measurement loaded when PERIOD = TRUE.
BUSY	BOOL	I, Q, M, D, L	Processing status
STATUS	WORD	I, Q, M, D, L	Status of the operation

Description

With the "Control high-speed counters" operation, you can make parameter settings and control the high-speed counters supported by the CPU by loading new values into the counter. The operation can only execute if a high-speed counter you want to control is enabled. It is not possible to insert or execute multiple "Control high-speed counters" operations simultaneously for each high-speed counter in the program.

You can load the following parameter values into a high-speed counter using the "Control high-speed counters" operation:

- Count direction (NEW_DIR): The count direction defines whether a high-speed counter counts up or down. The count direction is defined by the following values at the NEW_DIR input: 1 = up, -1 = down.
  A change to the count direction with the "Control high-speed counters" operation is possible only when direction control is set in the parameters by the program.
  The count direction specified at the NEW_DIR input is loaded into a high-speed counter when the bit at the DIR input is set.
- Count value (NEW_CV): The count value is the initial value at which a high-speed counter starts counting. The count value can be in a range from -2147483648 to 2147483647.
  The count value specified at the NEW_CV input is loaded into a high-speed counter when the bit at the CV input is set.
- Reference value (NEW_RV): You can compare the reference value with the current counter value to trigger an alarm. The reference value like the counter value can be in a range from -2147483648 to 2147483647.
  The reference value specified at the NEW_RV input is loaded into a high-speed counter when the bit at the RV input is set.
- Period of the frequency measurement (NEW_PERIOD): The period of the frequency measurement is specified by the following values at the NEW_PERIOD input: 10 = 0.01s, 100 = 0.1s, 1000 = 1s.
  The period can be updated when the "Frequency measurement" function is set in the parameters of the specified high-speed counter.
  The period specified at the NEW_PERIOD input is loaded into a high-speed counter if the bit at the PERIOD input is set.

The "Control high-speed counters" operation is only executed if the signal state at the EN input is "1". As long as the operation is executing, the bit at the BUSY output is set. Once the operation has executed completely, the bit at the BUSY output is reset.

The ENO enable output is set only when the EN input has signal state "1" and no errors occurred during execution of the operation.

'When inserting the "Control high-speed counters" operation, an instance data block is created in which the operation data is saved.

Parameter STATUS

At the STATUS output, you can query whether errors occurred during execution of the "Control high-speed counters" operation. The following table shows the meaning of the values output at the STATUS output:

Error code (hexadecimal)	Description
0	No error
80A1	Hardware identifier of the high-speed counter invalid
80B1	Count direction (NEW_DIR) invalid
80B2	Count value (NEW_CV) invalid
80B3	Reference value (NEW_RV) invalid
80B4	Period of the frequency measurement (NEW_PERIOD) invalid
80C0	Multiple access to the high-speed counter

9.8.5 Communication

9.8.5.1 S7 communication

Data consistency

Definition

The size of the data area which can be modified simultaneously by concurrent processes is called the consistent data area. Data areas which are larger than the consistent data area can thus be falsified as a whole.

This means that a data area which belongs together and which is larger than consistent data area can consist in part of new and of old consistent data blocks at the same time.

Example

An inconsistency can arise if a communication block is interrupted, for example, by a hardware interrupt OB with a higher priority. If the user program in this OB now changes the data which have already been processed in part by the communication block, the transferred data originate:

- In part from the time before the hardware interrupt was processed
- And in part from the time after the hardware interrupt was processed

This means that these data are inconsistent (not coherent).

Effect

If larger packages of data are to be transferred in a consistent form, the transfer should not be interrupted. This can, for example, increase the interrupt reaction time of the CPU.

In other words: The greater the quantity of data which must be transferred with absolute consistency, the longer the interrupt reaction time of a system.

Data consistency with SIMATIC

- If the user program contains a communication function that accesses common data, access to this data area can be coordinated, e.g., by means of the DONE parameter itself. The data consistency of the communication areas which are transferred locally with a communication block can therefore be ensured in the user program.
- In the case of S7 communication instructions "PUT (Page 1856)"/"GET (Page 1854)", the size of the consistent data areas must already be taken into consideration during programming or configuration, since no communication block is available in the user program of the target device (server) to synchronize communication data to the user program.
- These communication areas can then be accessed consistently using the "PUT (Page 1856)" / "GET (Page 1854)" instructions or when reading/writing tags, for example by an OP or an OS.

Note

Additional information on data consistency is provided in the description of the specific instructions.

Common parameters of instructions for S7 communication

Classification

The parameters of the instructions for S7 communication can be divided into the following five categories according to their functions:

1. Control parameters are used to activate an instruction.
2. Addressing parameters are used to address the remote communication partner.
3. Send parameters point to the data areas that are to be sent to the remote partner.
4. Receive parameters point to the data areas where the data received from remote partners will be entered.
5. Status parameters are used to monitor whether the instruction has completed its task without error or for the analysis of any errors that have occurred.

Control parameters

Data exchange will only be activated if the appropriate control parameters have a defined value (for example, are set) when the instruction is called or when the value has undergone a specific change since the previous call (for example, a positive edge).

Note

First call

For the first call, set the REQ parameter to FALSE .

Addressing parameter ID

The parameter ID is a reference to the local connection description (specified by the configuration of the connection).

Note

Addressing parameter ID

You can reassign the addressing parameter ID during runtime. The new parameter is validated with each new job after the previous job has been closed.

You can save instance DBs and therefore work memory if you use an instance data block for several connections via IDs.

Status parameters

With the status parameters, you monitor whether the instruction has completed its task correctly or whether it is still active. The status parameters also indicate errors.

Note

The status parameters are valid for one cycle only, namely from the first command following the call until the next call. As a result, you must evaluate these parameters after each instruction cycle.

Send and receive parameters

If you do not use all send or receive parameters of an instruction, the first unused parameter must be a NIL pointer and the parameters used must be located one after the other without any gaps.

For instructions for two-way communication

- The number of the SD and RD parameters used must match at the send and receive end.
- The data types of the SD and RD parameters that belong together must match at the send and receive end.
- The amount of data to be sent using the SD parameter must not exceed the area made available by the corresponding RD parameter.

If you do not keep to the rules above, this is indicated by ERROR = 1 and STATUS = 4.

User data size

With the "GET (Page 1854)" and "PUT (Page 1856)" instructions, the amount of data to be transmitted must not exceed a defined user data length. The maximum user data size depends on the communication partner.

The guaranteed minimum size of the user data for an instruction with 1-4 tags is 160 bytes.

Further information on restrictions on the user data size can be found in the technical data of the CPU.

GET: Read data from a remote CPU

Description

With this instruction, you can read data from a remote CPU.

The instruction is started on a positive edge at control input REQ. The relevant pointers to the areas to be read out (ADDR_i) are then sent to the partner CPU.

The remote partner returns the data.

The received data is copied to the configured receive areas (RD_i) at the next call.

Make sure that the areas defined with the parameters ADDR_i and RD_i match in terms of number, length, and data type. If you use a VARIANT pointer at the RD_i parameters that accesses a DB, the DB must always be specified (for example: P# DB10.DBX5.0 Byte 10). The DB must be generated to be compatible with S7-300/400.

Completion of this action is indicated by the status parameter NDR having the value "1".

Reading can only be activated again after the previous reading process has been completed.

The remote CPU can be in RUN or STOP mode.

Errors and warnings are output via ERROR and STATUS if access problems occurred while the data was being read or if the data type check results in an error.

Parameters

The following table shows the parameters of the "GET" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange on a rising edge.
ID	Input	CONN_PRG (WORD)	I, Q, M, D, L or constant	Addressing parameter ID, see also: Common parameters of instructions for S7 communication (Page 1852)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still executing. • 1: Job completed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: – 0000H: Neither warning nor error – <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	

Parameters	Declaration	Data type	Memory area	Description
ADDR_1	InOut	REMOTE	D	Pointers to the areas on the partner CPU that are to be read. Only absolute addressing is permitted for the data type REMOTE (example: P#DB10.DBX5.0 Byte 10).
ADDR_2	InOut	REMOTE		
ADDR_3	InOut	REMOTE		
ADDR_4	InOut	REMOTE		
RD_1	InOut	VARIANT	I, Q, M, D, L	Pointers to the areas on the local CPU in which the read data are entered. Only data types BOOL are permitted (for a bit field the address must be "0" and the length an integer multiple of byte), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL.
RD_2	InOut	VARIANT		
RD_3	InOut	VARIANT		
RD_4	InOut	VARIANT		

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

Parameters ERROR and STATUS

The following table contains all specific error information for the "GET" instruction that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	4	Errors in the receive area pointers RD_x relating to the data length or the data type.
1	8	Access error on the partner CPU.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "GET" • A global DB was specified instead of an instance DB • No instance DB found (loading a new instance DB from the PG).

ERROR	STATUS (decimal)	Explanation
1	20	• Maximum number of parallel jobs/instances exceeded • The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP). • Possible when first called
1	27	There is no function code on the CPU for this instruction.

Note

Data consistency

Data is received consistently if you read the part of the receive area RD_i currently being used completely before initiating another job.

PUT: Write data to a remote CPU

Description

With this instruction, you can write data to a remote CPU.

The instruction is started on a positive edge at control input REQ. The pointers to the areas to be written (ADDR_i) and the data (SD_i) are then sent to the partner CPU.

The remote partner saves the required data under the addresses supplied with the data and returns an execution acknowledgement.

Make sure that the areas defined with the parameters ADDR_i and SD_i match in terms of number, length, and data type. If you use a VARIANT pointer at the RD_i parameters that accesses a DB, the DB must always be specified (for example: P# DB10.DBX5.0 byte 10). The DB must be generated to be compatible with S7-300/400.

If no errors occur, this is indicated at the next instruction call with status parameter DONE = "1".

The writing process can only be activated again after the last job is complete.

The remote CPU can be in RUN or STOP mode.

Errors and warnings are output via ERROR and STATUS if access problems occurred while the data was being written or if the execution check results in an error.

Parameters

The following table shows the parameters of the "PUT" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange on a rising edge.
ID	Input	CONN_PRG (WORD)	I, Q, M, D, L or constant	Addressing parameter ID, see also: Common parameters of instructions for S7 communication (Page 1852)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing • 1: Job has been executed error-free.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: – 0000H: Neither warning nor error – <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 There is an error. STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	
ADDR_1	InOut	REMOTE	D	Pointers to the areas on the partner CPU to which the data will be written. Only absolute addressing is permitted for the data type REMOTE (example: P#DB10.DBX5.0 Byte 10).
ADDR_2	InOut	REMOTE		
ADDR_3	InOut	REMOTE		
ADDR_4	InOut	REMOTE		
SD_1	InOut	VARIANT	I, Q, M, D, L	Pointers to the areas on the local CPU which contain the data to be sent. Only data types BOOL are permitted (for a bit field the address must be "0" and the length an integer multiple of byte), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIMER. **Note:** If the VARIANT pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10).
SD_2	InOut	VARIANT		
SD_3	InOut	VARIANT		
SD_4	InOut	VARIANT		

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

Parameters ERROR and STATUS

The following table contains all specific error information for the "PUT" instruction that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	4	Errors in the send area pointers SD_i relating to the data length or the data type
1	8	Access error on the partner CPU.
1	10	Access to the local user memory not possible (for example, access to a deleted DB)
1	12	When the instruction was called • An instance DB was specified that does not belong to "PUT" • A global DB was specified instead of an instance DB • No instance DB found (loading a new instance DB from the PG).
1	20	• Maximum number of parallel jobs/instances exceeded • The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP). • Possible when first called
1	27	There is no function code on the CPU for this instruction.

Data consistency

When a send operation is activated (rising edge at REQ), the data to be sent from the send area SD_i is copied from the user program. After the block call, you can write to these areas without corrupting the current send data.

Note

The send operation is only complete when the DONE status parameter has the value "1".

9.8.5.2 Open User Communication

TSEND_C: Send data via Ethernet (TCP)

Description

The "TSEND_C" instruction is executed asynchronously and has the following functions:

- Setting up and establishing a communications connection:
  "TSEND_C" Sets up and establishes a TCP or ISO-on-TCP communications connection. After the connection is set up and established, it is automatically maintained and monitored by the CPU. The connection description specified at the CONNECT parameter is used to set up the communications connection.

  To establish a connection, the CONT parameter must be set to the value "1". If connection establishment is successful, the DONE parameter is set to "1" for one cycle. An existing connection is terminated and the setup connection is removed when the CPU goes to STOP mode. To set up and establish the connection again, you must execute "TSEND_C" again. For information on the number of possible communications connections, refer to the technical specifications for your CPU.
- Sending data via an existing communications connection:
  You specify the send area with the DATA parameter. This includes the address and the length of the data to be sent. Do not use a data area with the data type BOOL or Array of BOOL at the DATA parameter. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
- The send job is executed when a rising edge is detected at the REQ parameter. With the LEN parameter, you specify the maximum number of bytes sent with a send job. When sending data (rising edge at the REQ parameter), the CONT parameter must have the value "1" in order to establish or maintain a connection. The data to be sent must not be edited until the send job has completed. If the send job executes successfully, the DONE parameter is set to "1". Signal state "1" at the DONE parameter does not imply confirmation that the sent data has already been read by the communications partner.
- Terminating the communications connection:

  The communications connection is terminated when the CONT parameter is set to "0".

TSEND_C is executed again when the COM_RST parameter is set to "1". This terminates the existing communications connection and a new connection is established. If data is being transferred when it executes again, this can lead to a loss of data.

Parameters

The following table shows the parameters of the "TSEND_C" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Starts the send job on a rising edge.
CONT	Input	BOOL	I, Q, M, D, L	Controls the communications connection: • 0: Disconnect the communications connection • 1: Establish and maintain the communications connection When sending data (rising edge at the REQ parameter), the CONT parameter must have the value TRUE in order to establish or maintain a connection.
LEN	Input	UINT	I, Q, M, D, L or constant	Maximum number of bytes to be sent with the job . If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
CONNECT	InOut	TCON_Param	D	Pointer to the connection description See also: Parameters of communication connections (Page 403)
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the send area containing the address and the length of the data to be sent.
COM_RST	InOut	BOOL	I, Q, M, D, L	Restarts the instruction: • 0: Irrelevant • 1: Complete restart of the instruction causing an existing connection to be terminated and a new connection to be established.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job completed error-free
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

Parameters BUSY, DONE and ERROR

You can check the status of the job with the BUSY, DONE, ERROR, and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of "TSEND_C". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE, and ERROR parameters:

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job was stopped with an error. The cause of the error is specified in the STATUS parameter.
0	0	0	No new job was assigned.

Parameters ERROR and STATUS

ERROR	STATUS (W#16#...)	Description
0	0000	Job completed error-free
0	7000	No job processing active
0	7001	• Start execution of the job • Establish connection • Wait for connection partner
0	7002	Data are being sent
0	7003	Connection is terminated
0	7004	Connection established and monitored, no job processing active.
1	80A0	Group error for error codes 80A1 and 80A2.
1	80A1	• Connection or port already being used by user. • Communications error: – The specified connection has not yet been established. – The specified connection is being terminated. A transfer over this connection is not possible. – The interface is being re-initialized.
1	80A2	Local or remote port is being used by the system.
1	80A3	Attempt being made to terminate a non-existent connection.
1	80A4	IP address of the remote endpoint of the connection is invalid, in other words, it matches the IP address of the local partner.
1	80A7	Communications error: You called the COM_RST = 1 instruction before the send job was complete.

ERROR	STATUS (W#16#...)	Description
1	80B2	The CONNECT parameter points to a data block that was generated with the attribute Only store in load memory.
1	80B3	Inconsistent parameter assignment: Group error for error codes 80A0 to 80A2, 80A4, 80B4 to 80B9.
1	80B4	You have violated one or both of the following conditions for passive connection establishment (active_est = FALSE) when using the ISO-on-TCP protocol variant (connection_type = B#16#12): "local_tsap_id_len >= B#16#02", and/or "local_tsap_id[1] = B#16#E0".
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description.
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	8085	The LEN parameter is higher than the highest permitted value.
1	8086	The ID parameter within the CONNECT parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible.
1	8088	The value at the LEN parameter does not match the receive area set at the DATA parameter.
1	8089	The CONNECT parameter does not point to a data block.
1	8091	Maximum nesting depth exceeded.
1	809A	The CONNECT parameter points to a field that does not match the length of the connection description.
1	809B	The ID of the local device in the connection description does not match the CPU.
1	80C3	• All connection resources are in use. • A block with this ID is already being processed in a different priority group.
1	80C4	Temporary communications error: • The connection cannot be established at this time. • The interface is receiving new parameters or the connection is being established. • The configured connection is being removed by a "TDISCON" instruction. • The connection used is being terminated by a call with COM_RST= 1.
1	8722	CONNECT parameter: The source area is invalid. The area does not exist in the DB.
1	873A	CONNECT parameter: Access to the connection description is not possible (for example, DB does not exist).
1	877F	CONNECT parameter: Internal error.
1	8822	DATA parameter: Invalid source area, the area does not exist in the DB.
1	8824	DATA parameter: Area error in the VARIANT pointer.
1	8832	DATA parameter: The DB number is too large.
1	883A	CONNECT parameter: Access to entered connection data not possible (e.g. because the DB does not exist).
1	887F	DATA parameter: Internal error, e.g. illegal VARIANT reference.
1	893A	DATA parameter: Access to entered transmission range not possible (e.g. because the DB does not exist).

See also

TCON: Establish communication connection (Page 1868)

TRCV_C: Receive data via Ethernet (TCP)

Description

The "TRCV_C" instruction is executed asynchronously and has the following functions:

- Setting up and establishing a communications connection:

  "TRCV_C" sets up and establishes a TCP or ISO-on-TCP communications connection. After the connection is set up and established, it is automatically maintained and monitored by the CPU.

  The connection description specified at the CONNECT parameter is used to set up the communications connection. To establish a connection, the CONT parameter must be set to the value "1". If the connection establishment is successful, the DONE parameter is set to "1".

  An existing connection is terminated and the setup connection is removed when the CPU goes to STOP mode. To set up and establish the connection again, you must execute "TRCV_C" again.

  For information on the number of possible communications connections, refer to the technical specifications for your CPU.
- Receiving data via an existing communications connection:
  If the EN_R parameter is set to the value "1", receipt of data is enabled. When receiving data (rising edge at the EN_R parameter), the CONT parameter must have the value TRUE in order to establish or maintain a connection.
- The received data is entered in a receive area. You specify the length of the receive area depending on the protocol variant being used either with the LEN parameter (if LEN <> 0) or the length information of the DATA parameter (if LEN = 0). If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
- After data has been received successfully, the signal state at the DONE parameter is "1". If errors occur in the data transfer, the DONE parameter is set to "0".
- Terminating the communications connection:

  The communications connection is terminated when the CONT parameter is set to "0".

TRCV_C is executed again when the COM_RST parameter is set. This terminates the existing communications connection and a new connection is established. If data is being received when it executes again, this can lead to a loss of data.

Receive modes of TRCV_C

The following table shows how the received data is entered in the receive area.

Protocol variant	Availability of data in the receive area	connection_type parameter of the connection description	LEN parameter	RCVD_LEN parameter
TCP (Ad-hoc mode)	The data are immediately available.	B#16#11	65535	1 to 1472
TCP (Data receipt with specified length)	The data are available as soon as the data length specified at the LEN parameter was fully received.	B#16#11	1 to 8192	Identical to the value at the LEN parameter
ISO on TCP (protocol-controlled data transfer)	The data are available as soon as the data length specified at the LEN parameter was fully received.	B#16#12	1 to 8192	Identical to the value at the LEN parameter

TCP (ad-hoc mode)

The ad-hoc mode is only available with the TCP protocol variant. You set ad-hoc mode by assigning the value "65535" to the LEN parameter. The length of the receive area is defined by the pointer at the DATA parameter. The actual received data length is output at the RCVD_LEN parameter. A maximum of 1472 bytes can be received.

TCP (Data receipt with specified length)

You use the value of the LEN parameter to specify the length for the data receipt. The data specified at the DATA parameter is available in the receive area as soon as the length specified at the LEN parameter has been completely received.

ISO-on-TCP (protocol-controlled data transfer)

With the ISO-on-TCP protocol variant, data is transferred protocol-controlled. The receive area is defined by the LEN and DATA parameters.

Parameters

The following table shows the parameters of the "TRCV_C" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Receive enable
CONT	Input	BOOL	I, Q, M, D, L	Controls the communications connection: • 0: Automatically disconnect communications connection after data have been sent • 1: Establish and maintain the communications connection When receiving data (rising edge at the EN_R parameter), the CONT parameter must have the value TRUE in order to establish or maintain a connection.
LEN	Input	UINT	I, Q, M, D, L or constant	Maximum length of the data to be received. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
CONNECT	InOut	TCON_Param	D	Pointer to the connection description See also: Parameters of communication connections (Page 403)
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the receive area
COM_RST	InOut	BOOL	I, Q, M, D, L	Restarts the instruction: • 0: Irrelevant • 1: Complete restart of the instruction causing an existing connection to be terminated
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job completed error-free
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction
RCVD_LEN	Output	UINT	I, Q, M, D, L	Amount of data actually received in bytes

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

Parameters BUSY, DONE and ERROR

You can check the status of the job with the BUSY, DONE, ERROR, and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of "TRCV_C". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE, and ERROR parameters:

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job was stopped with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

Parameters ERROR and STATUS

ERROR	STATUS (W#16#...)	Description
0	0000	Job completed error-free
0	7000	No job processing active
0	7001	• Start execution of the job • Establish connection • Wait for connection partner
0	7002	Data is being received
0	7003	Connection is being terminated
0	7004	• Connection established and monitored • No job processing active
1	8085	• The LEN parameter is higher than the highest permitted value. • The value at the LEN or DATA parameter was changed after the first call.
1	8086	The ID parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible
1	8088	The value at the LEN parameter does not match the receive area set at the DATA parameter.
1	8089	The CONNECT parameter does not point to a data block.
1	8091	Maximum nesting depth exceeded.
1	809A	The CONNECT parameter points to a field that does not match the length of the connection description.
1	809B	The ID of the local device (local_device_id) in the connection description does not match the CPU.
1	80A0	Group error for error codes W#16#80A1 and W#16#80A2.

ERROR	STATUS (W#16#...)	Description
1	80A1	• Connection or port already being used by user. • Communications error: – The specified connection has not yet been established. – The specified connection is being terminated. Transfer over this connection is not possible. – The interface is being re-initialized.
1	80A2	Local or remote port is being used by the system.
1	80A3	• Attempt being made to re-establish an existing connection. • Attempt being made to terminate a non-existent connection.
1	80A4	IP address of the remote endpoint of the connection is invalid, in other words, it matches the IP address of the local partner.
1	80A7	Communications error: You called the COM_RST = 1 instruction before the send job was complete.
1	80B2	The CONNECT parameter points to a data block that was generated with the attribute Only store in load memory.
1	80B3	Inconsistent parameter assignment: Group error for error codes W#16#80A0 to W#16#80A2, W#16#80A4, W#16#80B4 to W#16#80B9.
1	80B4	You have violated one or both of the following conditions for passive connection establishment (active_est = FALSE) when using the ISO-on-TCP protocol variant (connection_type = B#16#12): "local_tsap_id_len >= B#16#02", and/or "local_tsap_id[1] = B#16#E0".
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description.
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	80C3	• All connection resources are in use. • A block with this ID is already being processed in a different priority group.
1	80C4	Temporary communications error: • The connection cannot be established at this time. • The interface is receiving new parameters or the connection is being established. • The configured connection is being removed by a "TDISCON" instruction. • The connection used is being terminated by a call with COM_RST= 1.
1	8722	Error in the CONNECT parameter: Invalid source area (area not declared in data block).
1	873A	Error in the CONNECT parameter: Access to connection description is not possible (no access to data block).
1	877F	Error in the CONNECT parameter: Internal fault
1	8922	DATA parameter: Invalid target area, the area does not exist in the DB.
1	8924	DATA parameter: Area error in the VARIANT pointer.
1	8932	DATA parameter: The DB number is too large.
1	893A	CONNECT parameter: Access to entered connection data not possible (e.g. because the DB does not exist).

ERROR	STATUS (W#16#...)	Description
1	897F	DATA parameter: Internal error, e.g. illegal VARIANT reference.
1	8A3A	DATA parameter: No access to the data area because, for example, the data block does not exist.

Note

Error messages of the instructions "TCON", "TRCV" and "TDISCON"

Internally, the TRV_C instruction uses the "TCON (Page 1868)", "TRCV (Page 1876)" and "TDISCON (Page 1871)" instructions. The error messages of these instructions are contained in the respective descriptions.

TCON: Establish communication connection

Description

With "TCON" you set up and establish a communications connection. After the connection is set up and established, it is automatically maintained and monitored by the CPU. "TCON" is executed asynchronously.

To set up the communications connection, the connection data specified for the CONNECT and ID parameters is used. To establish the connection, a rising edge must be detected at the REQ parameter. If the connection establishment is successful, the DONE parameter is set to "1".

Number of possible connections

For information on the number of possible communications connections, refer to the technical specifications for your CPU.

Connection with TCP and ISO-on-TCP

Both communication partners call the "TCON" instruction to set up and establish the communications connection. During parameter assignment, you specify which partner is the active communication end point and which is the passive one.

If the connection aborts, for example due to a line break or due to the remote communications partner, the active partner attempts to reestablish the configured connection. You do not have to call "TCON" again.

An existing connection is terminated and the set-up connection is removed when the "TDISCON (Page 1871)" instruction is executed or when the CPU changes to STOP mode. To set up and establish the connection again, you will need to execute "TCON" again.

Parameters

The following table shows the parameters of the "TCON" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Starts the job to establish the connection specified in the ID on a rising edge.
ID	Input	CONN_OUC (WORD)	L, D or constant	Reference to the assigned connection. Range of values: W#16#0001 to W#16#0FFF
CONNECT	InOut	TCON_Param	D	Pointer to the connection description See also: Parameters of communication connections (Page 403)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job completed error-free
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

Parameters BUSY, DONE and ERROR

You can check the status of the job with the BUSY, DONE, ERROR, and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of "TCON". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE, and ERROR parameters:

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job was stopped with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

Parameters ERROR and STATUS

ERROR	STATUS (W#16#...)	Explanation
0	0000	Connection was established successfully
0	7000	No job processing active
0	7001	Start job execution, establish connection.
0	7002	Connection is being established (REQ irrelevant).
1	8086	The ID parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible
1	8089	The CONNECT parameter does not point to a data block.
1	809A	The CONNECT parameter points to a field that does not match the length of the connection description.
1	809B	The ID of the local device in the connection description does not match the CPU.
1	80A0	Group error for error codes W#16#80A1 and W#16#80A2.
1	80A1	Connection or port already being used by user.
1	80A2	Local or remote port is being used by the system.
1	80A3	Attempt being made to re-establish an existing connection.
1	80A4	IP address of the remote endpoint of the connection is invalid, in other words, it matches the IP address of the local partner.
1	80A7	Communications error: You called "TDISCON (Page 1871)" before "TCON" had completed.
1	80B2	The CONNECT parameter points to a data block that was generated with the attribute Only store in load memory.
1	80B3	Inconsistent parameter assignment: Group error for error codes W#16#80A0 to W#16#80A2, W#16#80A4, W#16#80B4 to W#16#80B9.
1	80B4	You have violated one or more of the following conditions for passive connection establishment with the ISO-on-TCP protocol variant (connection_type = B#16#12): • local_tsap_id_len >= B#16#02 • local_tsap_id[1] = B#16#E0 • With local_tsap_id_len >= B#16#03, local_tsap_id[1] is an ASCII character. • local_tsap_id[1] is an ASCII character and local_tsap_id_len >= B#16#03.
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	80B8	Parameter in the local connection description and ID parameter are different.
1	80C3	All connection resources are in use.
1	80C4	Temporary communications error: • The connection cannot be established at this time. • The interface is currently receiving new parameters. • The configured connection is being removed by a "TDISCON (Page 1871)" instruction.

TDISCON: Terminate communication connection

Description

The "TDISCON" instruction terminates a communications connection from the CPU to a communication partner.

Functional description

"TDISCON" is an asynchronous instruction, in other words, its job processing extends over multiple calls. You start the job for terminating a connection by calling the "TDISCON" instruction with REQ = 1.

After "TDISCON" has been successfully executed, the ID specified for "TCON (Page 1868)" is no longer valid and cannot be used for sending or receiving.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions (see also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 1220)).

The following table shows the relationship between BUSY, DONE, and ERROR. Using this table, you can recognize the current status of "TDISCON" or when the establishment of the connection is completed.

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job was stopped with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Parameters

The following table shows the parameters of the instruction "TDISCON":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter REQUEST starts the job for terminating the connection specified by ID. The job starts at rising edge.
ID	Input	CONN_OUC (WORD)	L, D or constant	Reference to the connection to the remote partner or between the user program and the communications level of the operating system that is to be terminated. ID must be identical to the associated parameter ID in the local connection description. Range of values: W#16#0001 to W#16#0FFF

Parameter	Declaration	Data type	Memory area	Description
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job executed without error.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet completed. • BUSY = 0: The job is completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: An error has occurred during processing, STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Error information

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

Parameters ERROR and STATUS

ERROR	STATUS(W #16#...)	Explanation
0	0000	Connection terminated successfully
0	7000	No job processing active
0	7001	Start of job processing, connection being terminated
0	7002	Intermediate call (REQ irrelevant), connection being terminated
1	8086	The ID parameter is not in the permitted range
1	80A3	Attempt being made to terminate a non-existent connection
1	80C4	Temporary communications error: The interface is receiving new parameters or the connection is currently being established.

TSEND: Send data via communication connection

Description

With "TSEND", you send data over an existing communications connection. "TSEND" is executed asynchronously.

- You specify the send area with the DATA parameter. This includes the address and the length of the data to be sent. All data types except BOOL and Array of BOOL can be used for the data to be sent.
- The send job is executed when a rising edge is detected at the REQ parameter.
- With the LEN parameter, you specify the maximum number of bytes sent with a send job.
  - When data is transferred with TCP, the "TSEND" provides no information about the length of the data sent to "TRCV (Page 1876)".
  - When data is transferred with ISO-on-TCP, the length of the data sent is communicated to "TRCV (Page 1876)".
- The data to be sent must not be edited until the send job has completed. If the send job executes successfully, the DONE parameter is set to "1". Signal state "1" at the DONE parameter does not imply confirmation that the sent data has already been read out by the communications partner.

Parameters

The following table shows the parameters of the "TSEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Starts the send job on a rising edge.
ID	Input	CONN_OUC (WORD)	D, L or constant	Reference to the connection established with "TCON". Range of values: W#16#0001 to W#16#0FFF
LEN	Input	UINT	I, Q, M, D, L	Maximum number of bytes to be sent with the job . If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
DATA	InOut	VARIANT	I, Q, M, D	Pointer to the send area containing the address and the length of the data to be sent. The address references: • The process image input • The process image output • A bit memory address • A data block
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job completed error-free

Parameter	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

Parameters LEN and DATA

- With LEN = 0 , all the data specified with the DATA parameter is sent.
- If the number of bytes at the LEN parameter is higher than the length of the data to be sent that was defined with the DATA parameter, the error code 8088 is output at the STATUS parameter (see description of the STATUS parameter in the following).
- If LEN > 0, with elementary data types, the length of LEN must correspond to the length of the data to be sent in bytes. If the length of the data does not match for elementary data types, the data will not be sent and the error code 8088 is output at the STATUS parameter.
- If a structure (Struct) is referenced via the DATA parameter, LEN can be shorter than the structure. In this case, only the data up to the length of the LEN parameter is transferred.
- If an array is referenced via the DATA parameter, LEN can be shorter than the entire array. The length of LEN must, however, be an integer multiple of the length of a single array element. If not, the data will not be sent and the error code 8088 is output at the STATUS parameter.
- With data types STRING and WSTRING, all data is transferred if the parameter LEN = 0. With LEN > 0 the length must be at least the maximum number of bytes plus two additional bytes, which contain the length information. You will find more detailed information on the structure of the data types in: "Overview of the valid data types (Page 741)".

Parameters BUSY, DONE and ERROR

You can check the status of the job with the BUSY, DONE, ERROR, and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of "TSEND". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE, and ERROR parameters:

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job was stopped with an error. The cause of the error is specified in the STATUS parameter.
0	0	0	No new job was assigned.

Note

Because "TSEND" is executed asynchronously, you must keep the data in the send area consistent until the DONE parameter or the ERROR parameter changes to the value "1".

Parameters ERROR and STATUS

ERROR	STATUS (W#16#...)	Description
0	0000	Send job completed without error.
0	7000	No job processing active.
0	7001	Start of job execution, data is being sent. When processing this job, the operating system accesses the data in the DATA send area.
0	7002	Job executing (REQ irrelevant). When processing this job, the operating system accesses the data in the DATA send area.
1	8085	The LEN parameter is higher than the highest permitted value.
1	8086	The ID parameter is outside the permitted address range.
1	8088	The LEN parameter is greater than the area specified in DATA.
1	80A1	Communications error: • The specified connection has not yet been established. • The specified connection is being terminated. Transfer over this connection is not possible. • The interface is being re-initialized.
1	80C3	• A block with this ID is already being processed in a different priority group. • Internal lack of resources.
1	80C4	Temporary communications error: • The connection cannot be established to the partner at this time. • The interface is receiving new parameter settings or the connection is being established.

TRCV: Receive data via communication connection

Description

With "TRCV", you receive data over an existing communications connection. "TRCV" is executed asynchronously.

Receipt of data is enabled when the EN_R parameter is set to the value "1". The received data is entered in a receive area. You specify the length of the receive area depending on the protocol variant being used either with the LEN parameter (if LEN <> 0) or the length information of the DATA parameter (if LEN = 0).

After successful receipt of data, the NDR parameter is set to the value "1". You can query the amount of data actually received at the RCVD_LEN parameter.

Receive modes of "TRCV"

The following table shows how the received data is entered in the receive area.

Protocol variant	Availability of data in the receive area	connection_type parameter of the connection description	LEN parameter	RCVD_LEN parameter
TCP (Ad-hoc mode)	The data is immediately available.	B#16#11	65535	1 to 1472
TCP (Receipt of data with specified length)	The data is available as soon as the data length specified at the LEN parameter was fully received.	B#16#11	1 to 8192	Identical to the value at the LEN parameter
ISO on TCP (Protocol-controlled data transfer)	The data is available as soon as the data length specified at the LEN parameter was fully received.	B#16#12	• 1 to 1452, if a CP is used. • 1 to 8192, if no CP is used.	Identical to the value at the LEN parameter

TCP (Ad-hoc mode)

The ad-hoc mode is only available with the TCP protocol variant. You set ad-hoc mode by assigning the value "65535" to the LEN parameter. The length of the receive area is defined by the pointer at the DATA parameter. The actual received data length is output at the RCVD_LEN parameter. A maximum of 1472 bytes can be received.

TCP (Receipt of data with specified length)

You use the value of the LEN parameter to specify the length for the data receipt. The data specified at the DATA parameter is available in the receive area as soon as the length specified at the LEN parameter has been completely received.

ISO on TCP (Protocol-controlled data transfer)

With the ISO-on-TCP protocol variant, data is transferred protocol-controlled.

The receive area is defined by the LEN and DATA parameters.

Parameters

The following table shows the parameters of the "TRCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Receive enable
ID	Input	CONN_OUC (WORD)	D, L or constant	Reference to the connection established with "TCON". Value range: W#16#0001 to W#16#0FFF
LEN	Input	UINT	I, Q, M, D, L or constant	Length of the receive area in bytes. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
DATA	InOut	VARIANT	I, Q, M, D	Pointer to the receive area
NDR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job completed error-free
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction
RCVD_LEN	Output	UINT	I, Q, M, D, L	Amount of data actually received in bytes

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

Parameters LEN, DATA, and RCVD_LEN

- If LEN = 0, the received data is saved in the receive area specified at the DATA parameter. The number of bytes received is indicated at the RCVD_LEN parameter.
- If the length specified at the LEN parameter is greater than the length of the data received at the DATA parameter, the error code 8088 is output at the STATUS parameter (see description of the STATUS) parameter in the following.
- If LEN > 0, with elementary data types, the length of LEN must correspond to the length of the data to be sent in bytes. If the length of the data does not match for elementary data types, the data will not be received and the error code 8088 is output at the STATUS parameter.
- If a structure (Struct) is referenced via the DATA parameter, LEN can be shorter than the structure. In this case, only the data up to the length of the LEN parameter is transferred.
- If the DATA parameter references a data block with optimized access, the total length of the data to be received must be specified as the length for the LEN parameter. Alternatively, the LEN parameter is set to "0". If the length of the data does not match for elementary data types, the data will not be received and the error code 8088 is output at the STATUS parameter.
- If an array is referenced via the DATA parameter, the length specified at the LEN parameter can be shorter than the entire array. The length at the LEN parameter must, however, be an integer multiple of the length of a single array element. If not, the data will not be sent and the error code 8088 is output at the STATUS parameter.
- If a data type STRING is referenced via the DATA parameter, the length specified at the LEN parameter may not be >=1 and <=2.
- If a data type WSTRING is referenced via the DATA parameter, the length specified at the LEN parameter may not be >=1 and <=5.

Parameters BUSY, NDR, and ERROR

You can check the status of the execution with the BUSY, NDR, ERROR, and STATUS parameters. The BUSY parameter indicates the processing status. With the NDR parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of TRCV. The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, NDR, and ERROR parameters:

BUSY	NDR	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job was stopped with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

Note

Because "TRCV" is executed asynchronously, the data in the receive area is only consistent when the NDR parameter is set to the value "1".

Parameters ERROR and STATUS

ERROR	STATUS (W#16#...)	Explanation
0	0000	Job executed successfully. The current length of the received data is output at the RCVD_LEN parameter.
0	7000	Block not ready to receive.
0	7001	Block is ready to receive, receive job was activated.
0	7002	Interim call, the receive job is executing. Note: While the job is being processed, data is written to the receive area. This means that an error can lead to inconsistent data in the receive area.
1	8085	• The LEN parameter is higher than the highest permitted value. • The value of the LEN or DATA parameter was changed after the first call.
1	8086	The ID parameter is outside the permitted address range.
1	8088	• Receive area is too small. • The value at the LEN parameter is higher than the receive area set at the DATA parameter.
1	80A1	Communications error: • The specified connection has not yet been established. • The specified connection is being terminated. Receive job over this connection is not possible. • The connection is being re-initialized.
1	80B3	Inconsistent parameter assignment
1	80C3	• A block with this ID is already being processed in a different priority group. • Internal lack of resources.
1	80C4	Temporary communications error: • The connection cannot be established to the partner at this time. • The interface is receiving new parameter settings or the connection is being established.

Structure of the address information for the remote partner with UDP

Overview

- With "TUSEND (Page 1881)", you transfer the address information of the receiver at the ADDR parameter. This address information must have the structure specified below.
- With "TURCV (Page 1884)", you receive the address of the sender of the received data at the ADDRparameter. This address information must have the structure specified below.

Data block for the address information of the remote partner

You must create a DB that contains one or more data structures as per PLC data type (UDT) "TADDR_PAR".

In parameter ADDR of "TUSEND (Page 1881)", you transfer a pointer of type VARIANT to the address of the associated remote partner (e.g. P#DB100.DBX0.0 byte 8). This pointer is received in the ADDR parameter of "TURCV (Page 1884)".

Configuration of the address information for the remote partner for "TADDR_PAR" (UDT 66)

Byte	Parameter	Data type	Start value	Description
0 to 3	rem_ip_addr	ARRAY [1..4] of BYTE	B#16#00 ...	IP address of the remote partner, e.g. 192.168.002.003: • rem_ip_addr[1] = B#16#C0 (192) • rem_ip_addr[2] = B#16#A8 (168) • rem_ip_addr[3] = B#16#02 (002) • rem_ip_addr[4] = B#16#03 (003)
4 to 5	rem_port_nr	ARRAY [1..2] of BYTE	B#16#00 ...	remote port no. (possible values see: Assignment of port numbers (Page 406)): • rem_port_nr[1] = high byte of the port no. in hexadecimal notation • rem_port_nr[2] = low byte of port no. in hexadecimal notation
6 to 7	spare	ARRAY [1..2] of BYTE	B#16#00 ...	Standby: Assign "0" to this parameter.

TUSEND: Send data via Ethernet (UDP)

Description

The "TUSEND" instruction sends data to the remote partner addressed by the ADDR parameter using UDP.

⚠ WARNING

When transferring data with UDP according to RFC 768, the data is transferred to the remote partner without acknowledgement and is therefore unreliable. This means that data can be lost without this being indicated by the block.

Note

For sequential send operations to different partners, you only need to adjust the ADDR parameter when calling "TUSEND". You do not need to call the "TCON (Page 1868)" and "TDISCON (Page 1871)" instructions again.

Functional description

"The TUSEND" instruction works asynchronously; that is, its job processing extends over multiple calls. You start the job by calling "TUSEND" with REQ = 1.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 1220).

The following table shows the relationship between BUSY, DONE, and ERROR. Using this table, you can determine the current status of "TUSEND" or when the send process is concluded.

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job was stopped with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Note

Due to the asynchronous operation of "TUSEND", make sure the data in the send area remains consistent until the DONE parameter or the ERROR parameter has the value TRUE.

Parameters

The following table shows the parameters of the instruction "TUSEND":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter REQUEST starts the send job on a rising edge. The data is transferred from the area specified by DATA and LEN.
ID	Input	WORD	M, D or constant	Reference to the associated connection between the user program and the communication layer of the operating system. ID must be identical to the associated parameter ID in the local connection description. Range of values: W#16#0001 to W#16#0FFF
LEN	Input	UINT	I, Q, M, D, L	Number of bytes to be sent with the job Range of values: 1 to 1472
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job executed without error.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet completed. A new job cannot be triggered. • BUSY = 0: The job is completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: Error occurred during processing. STATUS supplies detailed information on the type of error
STATUS	Output	WORD	M, D	Status parameter STATUS: Error information
DATA	InOut	VARIANT	I, Q, M, D	Sender area, contains address and length The address refers to: • The process image input • The process image output • A bit memory address • A data block
ADDR	InOut	VARIANT	D	Pointers to the address of the receiver (for example, P#DB100.DBX0.0 byte 8) See also: Structure of the address information for the remote partner with UDP (Page 1880)

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

Parameters ERROR and STATUS

ERROR	STATUS(W#16#...)	Explanation
0	0000	Send job completed without error
0	7000	No job processing active
0	7001	Start of job processing, data being sent Note: During this processing phase, the operating system accesses the data in the DATA send area.
0	7002	Intermediate call (REQ irrelevant), job is being processed Note: During this processing phase, the operating system accesses the data in the DATA send area.
1	8085	The LEN parameter has the value "0" or is greater than the highest permitted value.
1	8086	The ID parameter is not in the permitted range
0	8088	The LEN parameter is greater than the memory area specified in DATA.
1	8089	The ADDR parameter does not point to a data block
1	80A1	Communications error: • The specified connection between user program and communication layer of the operating system has not yet been established. • The specified connection between the user program and the communication layer of the operating system is currently being terminated. Transmission over this connection is not possible. • The interface is being reinitialized.
1	80A4	IP address of the remote connection end point is invalid, e.g., it matches the local partner's own IP address.
1	80B3	• The protocol variant (connection_type parameter in the connection description) is not set to UDP. Please use "TSEND (Page 1873)". • ADDR parameter: Invalid information for port no.
1	80C3	• A block with this ID is already being processed in a different priority class. • Internal lack of resources.
1	80C4	Temporary communications error: • The connection between the user program and the communication layer of the operating system cannot be established at this time. • New parameter settings are being assigned to the interface.
1	8xyy	General error information: See also: Evaluating errors with output parameter RET_VAL (Page 1222)

TURCV: Receive data via Ethernet (UDP)

Description

The "TURCV" instruction receives data via UDP. After successful completion of "TURCV", the ADDR parameter will show you the address of the remote partner (the sender).

⚠ WARNING

When transferring data with UDP according to RFC 768, the data is transferred to the remote partner without acknowledgement and is therefore unreliable. This means that data can be lost without this being indicated by the block.

Functional description

"The TURCV" instruction works asynchronously; that is, its job processing extends over multiple calls. You start the receive job by calling "TURCV" with EN_R = 1.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 1220).

The following table shows the relationship between BUSY, NDR, and ERROR. Using this table, you can determine the current status of TURCV or when the receive process is concluded.

BUSY	NDR	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job was stopped with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Note

Due to the asynchronous operation of "TURCV", the data in the receive area is only consistent when the NDR parameter has the value TRUE.

Parameters

The following table shows the parameters of the instruction "TURCV":

Parameters	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter enabled to receive: With EN_R = 1, "TURCV" is ready to receive. The receive job is being processed.
ID	Input	WORD	M, D or constant	Reference to the associated connection between the user program and the communication layer of the operating system. ID must be identical to the associated parameter ID in the local connection description. Value range: W#16#0001 to W#16#0FFF
LEN	Input	UINT	I, Q, M, D, L	Length of the receive area in bytes: 0 (recommended) or 1 to 1472
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • NDR = 0: Job not yet started or still running. • NDR = 1: Job successfully completed
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: Error occurred during processing. STATUS supplies detailed information on the type of error
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet completed. A new job cannot be triggered. • BUSY = 0: The job is completed.
STATUS	Output	WORD	M, D	Status parameter STATUS: Error information
RCVD_LEN	Output	UINT	I, Q, M, D, L	Amount of data actually received, in bytes
DATA	InOut	VARIANT	I, Q, M, D	Receive area The address references: • The process image input • The process image output • A bit memory address • A data block
ADDR	InOut	VARIANT	D	Pointers to the address of the receiver (for example, P#DB100.DBX0.0 byte 8), see also: Structure of the address information for the remote partner with UDP (Page 1880)

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

Parameters ERROR and STATUS

ERROR	STATUS(W #16#...)	Explanation
0	0000	New data were accepted. The current length of the received data is shown in RCVD_LEN.
0	7000	Block not ready to receive
0	7001	Block is ready to receive, receive job was activated
0	7002	Intermediate call, receive job being processed Note: During this processing phase, "TURCV" writes data to the receive area. For this reason, an error could result in inconsistent data in the receive area.
1	8085	The LEN parameter is greater than the largest permitted value, or you changed the value of the LEN or DATA parameter since the first call
1	8086	The ID parameter is not in the permitted range
1	8088	• Receive area is too small. • Value in LEN is higher than the receive area specified by DATA
1	8089	The ADDR parameter does not point to a data block
1	80A1	Communications error: • The specified connection between user program and communication layer of the operating system has not yet been established. • The specified connection between the user program and the communication layer of the operating system is currently being terminated. A receive job over this connection is not possible. • New parameter settings are being assigned to the interface.
1	80B3	The protocol variant (connection_type parameter in the connection description) is not set to UDP. Please use "TRCV (Page 1876)".
1	80C3	• A block with this ID is already being processed in a different priority class. • Internal lack of resources.
1	80C4	Temporary communications error: New parameter settings are being assigned to the interface.
1	8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 1222)

See also

TCON: Establish communication connection (Page 1868)

TDISCON: Terminate communication connection (Page 1871)

T_CONFIG: Configure interface

Description T_CONFIG

Description

The "T_CONFIG" instruction is used for the program-controlled configuration of the integrated PROFINET interface of the CPU. The existing configuration data is overwritten.

You can make the following interface configuration settings:

- IP parameters: IP address, subnet mask, router address
- PROFINET IO device name (if the CPU is operated as a PROFINET IO device)

You store the configuration data in a data block (CONF_DB parameter).

You can make the program-controlled setting of the IP configuration with the "T_CONFIG" instruction as an alternative to configuration in the device configuration. It only takes effect, however, if you explicitly specified in the hardware configuration that IP parameters are assigned via the user program.

Functional description

The "T_CONFIG" instruction works asynchronously, that is, its execution extends over multiple calls. You start the transfer operation by calling "T_CONFIG" with REQ = 1. Only one job can be active at any time.

The job status is indicated by the output parameters BUSY and STATUS.

The following table shows the relationship between BUSY, DONE, and ERROR. Using this table, you can determine the current status of the instruction and when the transfer of configuration data is concluded.

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job was stopped with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Parameters

The following table shows the parameters of the "T_CONFIG" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	You start processing of the instruction when calling the instruction with REQ = 1. When you call the instruction with REQ = 0, the status displays DONE, ERROR and STATUS will be updated.
INTERFACE	Input	HW_INTERFACE	I, Q, M, D, L or constant	Hardware identification of the interface (see "Properties" in the Inspector window of the device configuration). The hardware ID is stored in the system constants of the PLC tags.
CONF_DATA (Page 1890)	InOut	VARIANT	D	Pointer to a data block in which you store the connection data. Use the pointer to reference a higher-level Struct element containing the Header, Addr and NOS fields as subelements (refer to the description of the CONF_DATA parameter).
DONE	Output	BOOL	I, Q, M, D, L	The status parameter indicates if the job was executed without errors: • 0: Processing not yet complete. • 1: Processing of instruction finished successfully.
BUSY	Output	BOOL	I, Q, M, D, L	Status of the instruction: • 0: Processing of the instruction has not started, completed or canceled yet. • 1: Processing of instruction in progress
ERROR	Output	BOOL	I, Q, M, D, L	Error display • 0: No error • 1: Error
STATUS	Output	DWORD	I, Q, M, D, L	Status display For meaning in connection with the parameters DONE and ERROR see under displays of the instruction.
ERR_LOC	Output	DWORD	I, Q, M, D, L	Error location (fieldId and id of the subfield in which an error has occurred at a parameter)

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

Parameters ERROR and STATUS

ERROR	STATUS (DW#16#..)	ERR_LOC*	Explanation
0	00000000	0	Job execution terminated without error
0	00700000	0	No job processing active
0	00700100	0	Start of job execution
0	00700200	0	Intermediate call (REQ irrelevant)
1	C08xyy00	0	General error information See also: Evaluating errors with output parameter RET_VAL (Page 1222)
1	C0808000	0	Hardware identification at parameter INTERFACE invalid.
1	C0808100	0	Hardware identification at parameter INTERFACE is not assigned to the supported PROFINET interface.
1	C0808700	0	Incorrect length of the data block at the CONF_DATA parameter.
1	C0808800	f, 0	Field_type has an illegal value.
1	C0808900	f, 0	The parameter fieldid has an illegal value or was used several times.
1	C0808A00	f, 0	Incorrect number for subfield_cnt parameter or incorrect length at Length parameter.
1	C0808B00	f, s	The parameter Id of a subfield contains an illegal value.
1	C0808C00	f, s	Error when placing subfield (incorrect subfield, incorrect sequence or multiple use of subfield).
1	C0808D00	f, s	The parameter Lenght of a subfield contains an incorrect or illegal value.
1	C0808E00	f, s	The parameter Mode of a subfield contains an incorrect or illegal value.
1	C0809000	f, s	The parameters of the subfield are write-protected. E.g. parameters are specified via configuration or PNIO mode is enabled.
1	C0809100	f, s	Reserved
1	C0809400	f, s	The parameter value in the subfield is not defined or illegal.
1	C0809500	f, s	The value of a subfield parameter is inconsistent with another parameter value.
1	C080C200	0	Transfer cannot be carried out (e.g. because the interface is not reachable).
1	C080C300	0	Insufficient resources (for example, multiple calling of "T_CONFIG" with different parameters)
1	C080C400	0	Temporary communications error
1	C080D200	0	Call not possible / not supported by the PROFINET interface
* In the table above, f is the field_id and s the id of the subfield in which the error occurred.			

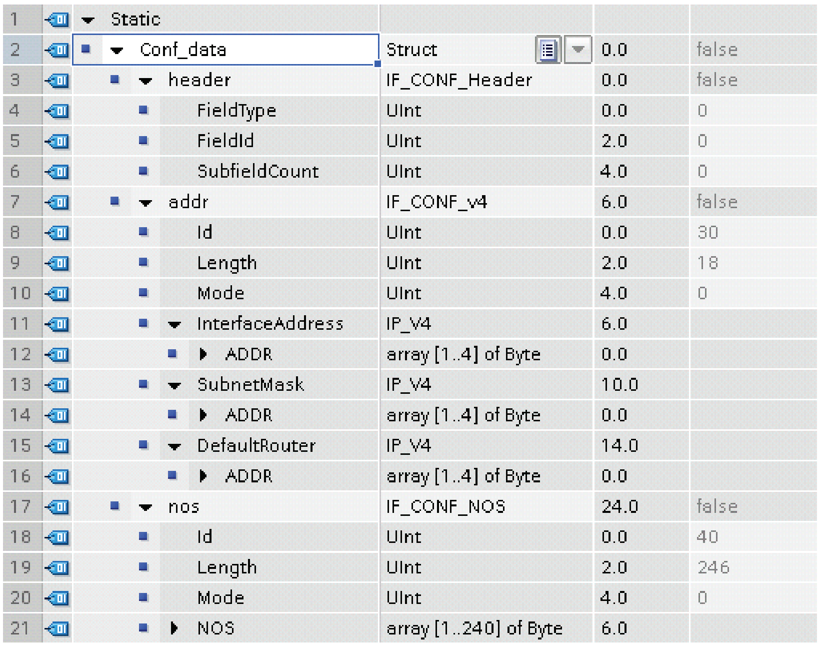
Parameter CONF_DATA

Structure of the DBs of the configuration data

The CONF_DATA parameter of the "T_CONFIG" instruction points to a global data block (DB), in which you store the configuration data.

The DB consists of the structure IF_CONF_Header and the structures IF_CONF_V4 and IF_CONF_NOS:

- The structure IF_CONF_Header has to be at the start of the DB. Use the structure to determine the number of subfields you want to use.
- The structures IF_CONF_V4 and IF_CONF_NOS are the subfields used in the DB which contain the actual configuration data. The respective parameters of the two subfields correspond largely to the structure in the device properties.
- All three structures must be defined below a higher-level structure (in the following example the Struct Element "Conf_Data"). The following figure shows the data block structure.



1	Static			
2	Conf_data	Struct	0.0	false
3	header	IF_CONF_Header	0.0	false
4	FieldType	UInt	0.0	0
5	FieldId	UInt	2.0	0
6	SubfieldCount	UInt	4.0	0
7	addr	IF_CONF_v4	6.0	false
8	Id	UInt	0.0	30
9	Length	UInt	2.0	18
10	Mode	UInt	4.0	0
11	InterfaceAddress	IP_V4	6.0	
12	ADDR	array [1..4] of Byte	0.0	
13	SubnetMask	IP_V4	10.0	
14	ADDR	array [1..4] of Byte	0.0	
15	DefaultRouter	IP_V4	14.0	
16	ADDR	array [1..4] of Byte	0.0	
17	nos	IF_CONF_NOS	24.0	false
18	Id	UInt	0.0	40
19	Length	UInt	2.0	246
20	Mode	UInt	4.0	0
21	NOS	array [1..240] of Byte	6.0	

Interconnection of the data block in the CONF_DATA parameter

In the CONF_DATA parameter, call the higher-level Struct element of the data block (in the example above the Struct element "Conf_Data"; the call in the parameter is achieved by specifying the data block followed by the name of the Struct element: "Name_of_DB".Conf_data).

Field IF_CONF_Header

Use the field IF_CONF_Header to select how many subfields you want to use during execution of "T_CONFIG".

Byte	Parameters	Data type	Start value	Description
0 ... 1	FieldType	UINT		Field type: Must always be 0.
2 ... 3	FieldId	UINT		Error ID: Must always be 0.
4 ... 5	SubfieldCount	UINT		Total number of subfields in the structure

General parameters of the subfields

The subfields "Addr" and "Nos" contain the following general parameters:

- Id

  This parameter identifies the respective field and may not be altered.
- Length

  This parameter specifies the actual length of the subfield. If a field contains a parameter of the data type String or Array, it may be that the maximum length of the parameter is not exhausted. In this case the actual length of the subfield is less than the maximum length.
- Mode

  The following values are permitted for this parameter:

  - 1: Permanent validity of the configuration data
  - 2: Temporary validity of the configuration data including the deletion of existing permanent configuration data

Subfield IF_CONF_V4

Use the subfield IF_CONF_V4 to specify the Ethernet addresses that you want to assign to the interface of the CPU.

Byte	Parameters	Data type	Start value	Description
0 ... 1	Id	UINT	30	Subfield identifier
2 ... 3	Length	UINT	18	Length of the subfield in bytes
4 ... 5	Mode	UINT		Validity of addressing: • 1: permanent • 2: temporary
6 ... 9	InterfaceAddress	IP_V4 *		IP address
10 ... 12	SubnetMask	IP_V4 *		Subnet mask
14 ... 16	DefaultRouter	IP_V4 *		Router address
* The data type IP_V4 is a structure of 4 BYTE, which includes the respective address of the respective parameter (e.g. at parameter SubnetMask the four-digit address of the subnet mask of the IP protocol).				

Subfield IF_CONF_NOS

Use the subfield IF_CONF_NOS to specify the station name to be assigned during execution of the instruction "T_CONFIG".

Byte	Parameters	Data type	Start value	Description
0 ... 1	Id	UINT	40	Subfield identifier
2 ... 3	Length	UINT	246	Length of the subfield in bytes
4 ... 5	Mode	UINT		Validity of the station name change: • 1: permanent • 2: temporary
6 ... 244	NoS	ARRAY [1...240] of Byte		Station name: You must occupy the ARRAY from the first byte. If the ARRAY is longer than the station name to be assigned, you must enter a zero byte after the actual station name (in conformity with IEC 61185-6-10). Otherwise, NoS is rejected and the "T_CONFIG" instruction enters the error code DW#16#C0809400 in STATUS. If you occupy the first byte with zero, the station name is deleted.

The station name is subject to the following limitations:

- Restricted to a total of 240 characters (lower case letters, numbers, dash, or dot)
- A name component within the station name, i.e., a character string between two dots, must not exceed 63 characters.
- No special characters such as umlauts, brackets, underscore, slash, blank space, etc. The only special character permitted is the dash.
- The station name must not begin or end with the "-" character.
- The station name must not begin with a number.
- The station name form n.n.n.n (n = 0, ... 999) is not permitted.
- The station name must not begin with the string "port-xyz" or "port-xyz-abcde" (a, b, c, d, e, x, y, z = 0, ... 9).

Note

You can also create an ARRAY NoS that is shorter than 240 bytes, but not less than 2 bytes. In this case, you must adjust the "Length" (length of subfield) tag accordingly.

9.8.5.3 Web server

WWW: Synchronizing user-defined web pages

Description

The instruction WWW initializes the Web server of the CPU or synchronizes user-defined web pages with the user program in the CPU.

User-defined web pages together with the web server make it possible for the CPU to access freely designed web pages of the CPU with a web browser.

Use script instructions (such as Javascript) and HTML code in user-defined web pages to transfer data via a web browser for further processing to the CPU and to display data from the operand area of the CPU in the web browser. Call the WWW instruction in the user program for synchronization of the user program and the web server as well as initialization.

Initialization

User-defined web pages are "packed" into data blocks so that the CPU can process them. You will have to generate data blocks from the source files (HTML files, screens, Javascript files, ...) during configuration. The Web Control DB takes on a special role (default: DB 333), it includes the status and control information as well as the references to other data blocks with coded web pages. The data blocks with coded web pages are called fragment DBs.

If the data blocks were loaded to the CPU, then the CPU does not "know" that the web pages in it are coded. The instruction "WWW" in the startup OB, for example, will inform the CPU which DB is the Web Control DB. The user-defined web pages will be available after this initialization via a web browser.

Synchronization

If you want the user program to interact with the user-defined web pages, then the instruction WWW must be used in the cyclical program part.

Examples for interaction between user program and web page:

- Checking received data
- Compiling and returning data for web browser making request

In this case it must be possible to evaluate the current status information and the web server must receive control information, such as release of a web page requested by a web browser.

Parameter

The following table shows the parameters of the instruction "WWW":

Parameter	Declaration	Data type	Description
CTRL_DB	Input	BLOCK_DB	Data block that describes the user-defined web pages (web control DB)
RET_VAL	Output	INT	Error information

For additional information on valid data types, refer to Overview of the valid data types (Page 741).

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred. There are no requests by web pages that have to be released by the user program.

00xy	x: indicates if an error has occurred during initialization of the web control DB (CTRL_DB):
	x=0: No errors occurred.
	x=1: Error occurred. The error is coded in the byte "CTRL_DB.last_error" of the web control DB, see description of web control DB.
	y: Number of the pending request. Several requests are possible (e. g. requests "0" and "1" are pending: y="3".
	y="1": Request "0"
	y="2": Request "1"
	y="4": Request "2"
	y="8": Request "3"

803A	The specified web control DB does not exist on the CPU.
8081	Incorrect version or incorrect format of the web control DB
80C1	There are no resources to initialize the web application.

See also

Overview of the valid data types (Page 741)

9.8.5.4 Communications processor

Point-to-point

PORT_CFG: Configure communication parameters dynamically

Description

The instruction "PORT_CFG" allows dynamic configuration of communications parameters for a point-to-point communications port.

You set up the original static configuration of the port in the hardware configuration. You can change this configuration by executing the "PORT_CFG" instruction. You can also use this function to save created blocks in libraries and to avoid configuration in the hardware configuration when you reuse it.

With "PORT_CFG" you can influence the following communications parameter settings:

- Parity
- Baud rate
- Number of bits per character
- Number of stop bits
- Type and properties of flow control

The changes made by the "PORT_CFG" instruction are not stored permanently on the target system.

You can transfer serial data via the electrical connections RS-232 (half and full duplex) and RS-485 (half duplex).

Parameters

The following table shows the parameters of the "PORT_CFG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Activates the configuration change on a rising edge
PORT	Input	PORT (UINT)	D, L or constant	Identification of the communication port (HW-ID)
PROTOCOL	Input	UINT	I, Q, M, D, L or constant	Transmission protocol: • 0: Point-to-point communication protocol • 1..n: Future definition for specific transmission protocols
BAUD	Input	UINT	I, Q, M, D, L or constant	Baud rate of the port: • 1: 300 baud • 2: 600 baud • 3: 1200 baud • 4: 2400 baud • 5: 4800 baud • 6: 9600 baud (default) • 7: 19200 baud • 8: 38400 baud • 9: 57600 baud • 10: 76800 baud • 11: 115200 baud
PARITY	Input	UINT	I, Q, M, D, L or constant	Parity of the port: • 1: No parity (default) • 2: Even parity • 3: Odd parity • 4: Mark parity • 5: Space parity
DATABITS	Input	UINT	I, Q, M, D, L or constant	Bits per character: • 1: 8 bits per character (default) • 2: 7 bits per character
STOPBITS	Input	UINT	I, Q, M, D, L or constant	Number of stop bits: • 1: 1 stop bit (default) • 2: 2 stop bits
FLOWCTRL	Input	UINT	I, Q, M, D, L or constant	Data flow control: • 1: None (default) • 2: XON/XOFF • 3: Hardware flow control (RTS always activated) • 4: Hardware flow control (RTS can be deactivated during transmission)

Parameter	Declaration	Data type	Memory area	Description
XONCHAR	Input	CHAR	D	Indicates the character used as XON character. The character DC1 (11H) is set as default.
XOFFCHAR	Input	CHAR	D	Indicates the character used as XOFF character. The character DC3 (13H) is set as default.
WAITIME	Input	UINT	I, Q, M, D, L or constant	Specifies the wait time for XON or CTS after the start of the transmission. The specified value must be greater than 0. 2000 milliseconds are set as default.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job completed error-free
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

STATUS parameter

Error code (W#16#...)	Description
80A0	The specified protocol is invalid.
80A1	The specified baud rate is invalid.
80A2	The specified parity rate is invalid.
80A3	The specified number of bits per character is invalid.
80A4	The specified number of stop bits is invalid.
80A5	The specified type of flow control is invalid.
80A6	Incorrect value at the WAITTIME parameter When the data flow control is enabled, the value at the WAITTIME parameter must be greater than zero.
80A7	Invalid values at XONCHAR and XOFFCHAR parameters.

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 1913)".

SEND_CFG: Configure serial transmission parameters dynamically

Description

The instruction "SEND_CFG" allows dynamic configuration of serial transmission parameters for a point-to-point communications port. All the messages waiting for transfer are discarded after execution of SEND_CFG.

You set up the original static configuration of the port in the hardware configuration. You can change this configuration by executing the "SEND_CFG" instruction. You can also use this function to save created blocks in libraries and to avoid configuration in the hardware configuration when you reuse it. With "SEND_CFG" you can influence the following transmission parameter settings:

- Time between the activation of RTS (Request to Send) and the start of the transmission
- Time between the end of transmission and the deactivation of RTS
- Define bit times for breaks

The changes made by the "SEND_CFG" instruction are not stored permanently on the target system.

You can transfer serial data via the electrical connections RS-232 (half and full duplex) and RS-485 (half duplex).

Parameters

The following table shows the parameters of the "SEND_CFG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Activates the configuration change on a rising edge
PORT	Input	PORT (UINT)	D, L or constant	Identification of the communication port (HW-ID)
RTSONDLY	Input	UINT	I, Q, M, D, L or constant	The time that should elapse after activating RTS until the start of transmission. Valid values for this parameter are as follows: • 0 (default) • 0 to 65535 ms in steps of 1 ms This parameter does not apply to RS-485 modules.
RTSOFFDLY	Input	UINT	I, Q, M, D, L or constant	Time that should elapse after the end of transmission until deactivation of RTS. Valid values for this parameter are as follows: • 0 (default) • 0 to 65535 ms in steps of 1 ms This parameter does not apply to RS-485 modules.
BREAK	Input	UINT	I, Q, M, D, L or constant	Specifies the bit times for a break, which are sent at the start of the message. 12 bit times are set as default. A maximum of 25000 bit times can be specified.

Parameter	Declaration	Data type	Memory area	Description
IDLELINE	Input	UINT	I, Q, M, D, L or constant	Specifies the bit times for idle line after the break at the start of the message. 12 bit times are set as default. A maximum of 25000 bit times can be specified.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job completed error-free
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

STATUS parameter

Error code (W#16#...)	Description
80B0	The configuration of a transmission interruption is not permitted.
80B1	The specified break time exceeds the permitted maximum of 25000 bit times.
80B2	The specified time for idle line exceeds the permitted maximum of 25000 bit times.

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 1913)".

RCV_CFG: Configure serial receive parameters dynamically

Description

The instruction "RCV_CFG" allows dynamic configuration of serial receive parameters for a point-to-point communications port. You can use this instruction to configure the conditions that specify the start and end of the message to be transmitted. The receipt of messages that correspond to these conditions can be enabled by the "RCV_PTP (Page 1908)" instruction.

You set up the original static configuration of the port in the properties of the hardware configuration. Execute the "RCV_CFG" instruction in your program to change the configuration. You can also use this function to save created blocks in libraries and to avoid configuration in the hardware configuration when you reuse it. The changes made by the "RCV_CFG" instruction are not stored permanently on the target system.

All the messages waiting for transfer are discarded after execution of the "RCV_CFG" instruction.

Parameters

The following table shows the parameters of the "RCV_CFG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Activates the configuration change on a rising edge.
PORT	Input	PORT (UINT)	D, L or constant	Identification of the communication port (HW-ID)
CONDITIONS	Input	CONDITIONS	D, L	Data structure defining the conditions for start and end of data transmission.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job completed error-free
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

Data type CONDITIONS

You can use the CONDITIONS structure to define the start and end conditions for the message transmission. The structure CONDITIONS is included in the instance DB of the "RCV_CFG" instruction. Use the structure CONDITIONS to define the start and end conditions when the transmission of a message is complete and when the next message transfer is to start:

- You define the start conditions for the data transfer in the START structure.
- You define the end conditions for the data transfer in the END structure.

You can define one or more start and end conditions for this. If you specify multiple start or end conditions these are linked by an OR logic instruction.

The following table shows the "CONDITIONS" structure:

Parameters		Data type	Description
START		STRUCT	Start conditions
	STARTCOND	UINT	Specifies the start condition (details, see below). The start condition can be specified as a 16-bit hexadecimal value. Possible values for the start condition are: • 1: Start character • 2: Any character (default) • 4: Line break • 8: Idle line • 16: Character string 1 • 32: Character string 2 • 64: Character string 3 • 128: Character string 4 Multiple start conditions can also be defined at the STARTCOND parameter. The total from the values of the individual conditions is specified for this. If, for example, you want to define "Idle line" OR "Character string 1" OR "Character string 4" as start condition, the value "152" must be specified.
	IDLETIME	UINT	Specifies the maximum idle time of the line before receipt is started. Valid values for this parameter are as follows: • 40 bit times (default) • 0 to 2500 bit times
	STARTCHAR	BYTE	Specifies the start character. This setting is only enabled when the configured start condition is "Start character". Valid values for this parameter are as follows: • 02 (STX): Default setting • B#16#00 to B#16#FF

Parameters		Data type	Description
	SEQ[1].CTL	BYTE	Character string 1: Control sequence for each character You can use the bit position of the character to define which characters of the character string will be considered or ignored. To evaluate the characters, the corresponding bits have to be set. • Bit 0: 1 character • Bit 1: 2 characters • Bit 2: 3 characters • Bit 3: 4 characters • Bit 4: 5 characters A character is ignored when the corresponding bit is reset.
	SEQ[1].STR	CHAR[5]	Character string 1: Start character (5 characters)
	SEQ[2].CTL	BYTE	Character string 2: Ignore/compare control sequence for each character
	SEQ[2].STR	CHAR[5]	Character string 2: Start character (5 characters)
	SEQ[3].CTL	BYTE	Character string 3: Ignore/compare control sequence for each character
	SEQ[3].STR	CHAR[5]	Character string 3: Start character (5 characters)
	SEQ[4].CTL	BYTE	Character string 4: Ignore/compare control sequence for each character
	SEQ[4].STR	CHAR[5]	Character string 4: Start character (5 characters)
END		STRUCT	End conditions
	ENDCOND	UINT	Specifies the end condition (details, see below). The end condition can be specified as a 16-bit hexadecimal value. Possible values for the end condition are: • 1: Reply timeout • 2: Message timeout • 4: Timeout within the character string • 8: Maximum length • 16: N+LEN+M; the information on the message length in integrated in the message and will be evaluated. • 32: Character string 1 Multiple end conditions can also be defined at the ENDCOND parameter. The total from the values of the individual end conditions is specified for this. If, for example, you want to define the end condition "Maximum length" OR "Sequence 1", the value "40" must be specified.
	MAXLEN	UINT	Specifies the maximum number of characters in a message. Valid values* for this parameter are as follows: • 1 character (default) • 0 to 1024 characters This setting is only enabled if the "Maximum length" end condition is set at the ENDCOND parameter.
	N	UINT	Offset of the length field in the message Valid values for this parameter are as follows: • 0 characters (default) • 0 to 1024 characters This setting is only enabled if the "N+LEN+M" end condition is set at the ENDCOND parameter.

Parameters		Data type	Description
	LENGTHSIZE	UINT	Size of the length field in bytes Valid values* for this parameter are as follows: • 0 bytes (default) • 1 byte • 2 bytes • 4 bytes This setting is only enabled if the "N+LEN+M" end condition is set at the ENDCOND parameter.
	LENGTHM	UINT	Specifies the number of end characters that follow the length field but are not contained in the length of the message. Valid values for this parameter are as follows: • 0 characters (default) • 0 to 255 characters This setting is only enabled if the "N+LEN+M" end condition is set at the ENDCOND parameter.
	RCVTIME	UINT	Specifies the maximum duration for the receipt of the first character of a message. Valid values for this parameter are as follows: • 200 ms (default) • 0 to 65535 ms in steps of 1 ms This setting is only enabled if the "Reply timeout" end condition is set at the ENDCOND parameter.
	MSGTIME	UINT	Specifies the maximum duration of the receipt of a message. Valid values for this parameter are as follows: • 200 ms (default) • 0 to 65535 ms in steps of 1 ms This setting is only enabled if the "Message timeout" end condition is set at the ENDCOND parameter.
	CHARGAP	UINT	Specifies the time interval between received consecutive characters. Valid values for this parameter are as follows: • 12 bit times (default) • 0 to 2500 bit times This setting is only enabled if the "Timeout within the character string" end condition is set at the ENDCOND parameter.
	SEQ.CTL	BYTE	Character string: Control sequence for each character You can use the bit position of the character to define which characters of the character string will be considered or ignored. To evaluate the characters, the corresponding bits have to be set. • Bit 0: 1 character • Bit 1: 2 characters • Bit 2: 3 characters • Bit 3: 4 characters • Bit 4: 5 characters A character is ignored when the corresponding bit is reset.

Parameters		Data type	Description
	SEQ.STR	CHAR[5]	Character string: Start character (5 characters)
	* These value ranges also apply to the corresponding hardware settings for specifying the end of message.		

Start conditions for the message receipt (STARTCOND parameter)

The start of the message is recognized by the receiver if a configured start condition applies. The following conditions can be defined as start conditions for message receipt:

- Start character: The start of a message is recognized when a certain character occurs. This character is stored as first character of the message. All characters received before the start character are rejected.
- Any character: Any character can define the start of a message. This character is stored as first character of the message.
- Line break: The start of a message is recognized if the received data stream is interrupted for longer than one character length.
- Idle line: The start of a message is recognized when the send transmission line is in the idle state for a certain time (specified in bit times) followed by renewed transmission of characters.
- Character string (sequence): The start of a message is recognized when a specified character sequence occurs in the data stream. You can specify up to four character sequences with up to five characters each.

  Example: A received hexadecimal message includes the following characters: "68 10 aa 68 bb 10 aa 16". The configured start character sequences are listed in the following

  table. Start character sequences will be evaluated once the first character 68H

  has been received successfully. After the fourth character has been received successfully (the

  second 68H), the start condition "1" has been met. Once the start conditions have been met,

  evaluation of the end conditions will start.

  Processing of the start character sequence can end due to different errors in parity,

  framing or time intervals between characters. These errors will prevent

  reception of the message, because the start condition has not

  been met.

Start condition	First character	First character +1	First character +2	First character +3	First character +4
1	68H	xx	xx	68H	xx
2	10H	aaH	xx	xx	xx
3	dcH	aaH	xx	xx	xx
4	e5H	xx	xx	xx	xx

End conditions for the message receipt (ENDCOND parameter)

The start of a message is recognized by the receiver if a configured end condition applies. The following conditions can be defined as end conditions for message receipt:

- Reply timeout: The receipt of messages will end when the specified maximum duration for the receipt of a character is exceeded. The maximum duration is defined at the RCVTIME parameter. The defined time starts to run down as soon at the last transmission is completed and the RCV_PTP instruction enables the receipt of the message. If no character was received within the defined time (RCVTIME), the RCV_PTP instruction reports an error.
- Message timeout: The receipt of messages will end when the specified maximum duration for the receipt of a message is exceeded. The maximum duration is defined at the MSGTIME parameter. The defined time starts to run down as soon as the first character of the message is received.
- Timeout within the character string: The receipt of messages will end when the time interval between the receipt of two consecutive characters is longer than the value at the CHARGAP parameter.
- Maximum length: The receipt of messages will end when the length of the message defined at the MAXLEN parameter is exceeded.
- Reading message length (N+LEN+M): The receipt of messages will end when a certain message length is reached. This length is calculated by the values of the following parameters:
  - N: Position of the character in the message from which the length field begins.
  - LENGTHSIZE: Size of the length field in bytes
  - LENGTHM: Number of end characters that follow the length field. These characters are not taken into account in the evaluation of the message length.
- Character string: The receipt of messages will end when a defined character sequence is received. The character string can contain a maximum of five characters. For each character of the character string, you can use the bit position to define if this will be considered or ignored in the evaluation.

STATUS parameter

Error code (W#16#...)	Description
80C0	Error in start condition
80C1	• Error in end condition • No end condition defined
80C2	Receive interrupt enabled
80C3	A value that is equal to 0 or greater than 4132 was entered at the MAXLEN parameter while the "Maximum length" end condition was set.
80C4	A value that is greater than 4131 was entered at the N parameter while the "N+LEN+M" end condition was set.

Error code (W#16#...)	Description
80C5	A value that is equal to 0 or invalid was entered at the LENGTHSIZE parameter while the "N+LEN+M" end condition was set.
80C6	A value that is greater than 255 was entered at the LENGTHM parameter while the "N+LEN+M" end condition was set.
80C7	A message length greater than 4132 was calculated while the "N+LEN+M" end condition was set.
80C8	A value that is equal to 0 was entered at the RCVTIME parameter while the "Reply timeout" end condition was set.
80C9	A value that is equal to 0 or greater than 2500 was entered at the CHARGAP parameter while the "Timeout within a character string" end condition was set.
80CA	A value that is equal to 0 or greater than 2500 was entered at the IDLETIME parameter while the "Idle line" start condition was set.
80CB	All characters of the character string are marked as "Don't care" even though "Character string" is set as the end condition.
80CC	All characters of the character string are marked as "Don't care" even though "Character string" is set as the start condition.

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 1913)".

SEND_PTP: Transmit send buffer data

Description

You use the "SEND_PTP" instruction to start the transmission of data. The "SEND_PTP" instruction does not execute the actual transmission of the data. The data of the send buffer is transmitted to the relevant point-to-point communication module (CM). The CM executes the actual transmission.

Parameters

The following table shows the parameters of the "SEND_PTP" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Enables the requested transmission on a rising edge of this enable input. The content of buffer is transmitted to the point-to-point communication module (CM).
PORT	Input	PORT (UINT)	D, L or constant	Identification of the communication port (HW-ID)
BUFFER	Input	VARIANT	I, Q, M, D, L or constant	Pointer to the start address of the send buffer. Boolean values or Array of BOOL are not supported.
LENGTH	Input	UINT	I, Q, M, D, L or constant	Length of the send buffer

Parameter	Declaration	Data type	Memory area	Description
PTRCL	Input	BOOL	I, Q, M, D, L or constant	This parameter selects the buffer for normal point-to-point communication or for specific Siemens protocols implemented in the connected CM. FALSE = point-to-point operations controlled by the user program (only valid option)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job completed error-free
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

STATUS parameter

Error code (W#16#...)	Description
7000	The send operation is not active.
7001	The send operation is processing the first call.
7002	The send operation is processing subsequent calls (queries following the first call).
8080	The identifier entered for the communications port number is invalid.
8088	The length of the LENGHT parameter does not correspond to the length of data to be sent. See also: Parameters LENGHT and BUFFER.
80D0	A new send request was received while a transmission was taking place.
80D1	The transmission was interrupted because the CTS signal was not confirmed within the specified wait time.
80D2	The send request was interrupted because the communications partner (DCE) signaled that it was not willing to receive (DSR).
80D3	The send request was interrupted because the maximum size of the waiting loop was exceeded (more than 1024 Byte).

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 1913)".

Parameters LENGTH and BUFFER

The minimum data size that can be sent by the "PTP_SEND" instruction is one byte. The parameter BUFFER defines the size of the data to be sent. You can use neither the BOOL nor the Array of BOOL data type for the BUFFER parameter.

LENGTH parameter	BUFFER parameter	Description
LENGTH = 0	Not used	The complete data is sent as defined by the BUFFER parameter. If LENGTH = 0, you do not need to specify the number of bytes transferred.
LENGTH > 0	Elementary data type	The LENGTH value must contain the byte count of this data type. Otherwise, data is not transferred and error 8088 is output.
	STRUCT	The LENGTH value can contain a byte count that is smaller than the complete byte length of the structure. In this case, only the first LENGTH bytes are transferred.
	ARRAY	The LENGTH value can contain a byte count that is smaller than the complete byte length of the field. In this case, only the field elements that fit completely in the LENGTH bytes are transferred. The LENGTH value must be a multiple of the byte count of the data elements. Otherwise, STATUS = 8088, ERROR = 1, and no data is transferred.
	STRING	The complete memory arrangement of the character sequence format will be transmitted as well as the information about maximum length of the character string and the actual length of the character string. The LENGTH value must contain bytes for maximum length, actual length, and the characters of the character string. With the data type STRING, all lengths and characters have the size of one byte. If a character string is used for the BUFFER parameter, the LENGTH value must also contain two bytes for the two length fields.

RCV_PTP: Enable receive messages

Description

With the RCV_PTP instruction you enable receipt of a sent message. Each message must be enabled individually. The sent data is only available in the receive area when the message has been acknowledged by the relevant communications partner.

Parameters

The following table shows the parameters of the "RCV_PTP" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Enables receipt on a rising edge
PORT	Input	PORT (UINT)	D, L or constant	Identification of the communication port (HW-ID)
BUFFER	Input	VARIANT	I, Q, M, D, L or constant	Points to the start address of the receive buffer. Do not use a tag of the type STRING in the receive buffer.
NDR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job completed error-free
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction
LENGTH	Output	UINT	I, Q, M, D, L	Length of the message in the receive buffer

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

STATUS parameter

Error code (W#16#....)	Description
80E0	Receipt of messages was terminated because the receive buffer is full.
80E1	Receipt of messages was terminated as a result of a parity error.
80E2	Receipt of messages was terminated as a result of a framing error.
80E3	Receipt of messages was terminated as a result of an overflow error.
80E4	Receipt of messages was terminated because the calculated message length (N+LEN+M) exceeds the size of the receive buffer.
8080	The identifier entered for the communications port number is invalid.
8088	A data type STRING is referenced via the BUFFER parameter.
0094	Receipt of messages was terminated because the maximum character length was received.
0095	Receipt of messages was terminated as a result of a timeout.
0096	Receipt of messages was terminated because of a timeout within the character string.
0097	Receipt of messages was terminated as a result of a reply timeout.
0098	Receipt of messages was terminated because the "N+LEN+M" length condition has been satisfied.
0099	Receipt of messages was terminated because the character string defined as the end condition was received.

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 1913)".

RCV_RST: Delete receive buffer

Description

With the "RCV_RST" instruction, you delete the receive buffer of a communications partner.

Parameters

The following table shows the parameters of the "RCV_RST" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Enables deleting of the receive buffer on a rising edge
PORT	Input	PORT (UINT)	D, L or constant	Identification of the communication port (HW-ID)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job completed error-free
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 1913)".

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

SGN_GET: Query RS-232 signals

Description

With the "SGN_GET" instruction, you query the current state of several signals of an RS-232 communications module.

Parameters

The following table shows the parameters of the "SGN_GET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Enables the query on a rising edge
PORT	Input	PORT (UINT)	D, L or constant	Identification of the communication port (HW-ID)
NDR	Output	BOOL	I, Q, M, D, L	Is set for one cycle if new data are ready for sending and the instruction was executed error-free.
DTR	Output	BOOL	I, Q, M, D, L	Data terminal ready, module ready
DSR	Output	BOOL	I, Q, M, D, L	Data set ready, communications partner ready
RTS	Output	BOOL	I, Q, M, D, L	Send request, module ready to send
CTS	Output	BOOL	I, Q, M, D, L	Clear to send, communications partner can receive data (reaction to RTS = ON of the module).
DCD	Output	BOOL	I, Q, M, D, L	Data carrier detect, received signal level
RING	Output	BOOL	I, Q, M, D, L	Ring display, display of an incoming call
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

STATUS parameter

Error code (W#16#....)	Description
80F0	The communication module is an RS-485 module and no signals are available.

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 1913)".

SGN_SET: Set RS-232 signals

Description

With the "SGN_SET" instruction, you set the status of the output signals of an RS-232 communications module.

Parameters

The following table shows the parameters of the "SGN_SET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Activates the action on a rising edge Initial value: FALSE
PORT	Input	PORT (UINT)	D, L or constant	Identification of the communication port (HW-ID) Initial value: 0
SIGNAL	Input	BYTE	I, Q, M, D, L or constant	Specifies the signals to be set: • Set 01H = RTS • Set 02H = DTR • Set 04H = DSR Initial value: FALSE
RTS	Input	BOOL	I, Q, M, D, L or constant	Send request, module ready to send Initial value: FALSE
DTR	Input	BOOL	I, Q, M, D, L or constant	Data terminal ready, module ready Initial value: FALSE
DSR	Input	BOOL	I, Q, M, D, L or constant	Data set ready (applies only to interfaces of the DCE type) Initial value: FALSE
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job completed error-free Initial value: FALSE
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred Initial value: FALSE
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction Initial value: 0

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

STATUS parameter

Error code(W#16#...)	Description
80F0	The communication module is an RS-485 module and no signals are available.
80F1	No signals are settable because H/W flow control is enabled.
80F2	The DSR signal cannot be set because the module is a DTE device.
80F3	The DTR signal cannot be set because the module is a DCE device.

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 1913)".

General status information of the communications blocks

General information on execution status of the communications blocks

The following table shows which general information can be output at the STATUS parameter of the communications blocks:

Error code (W#16#....)	Description
8070	All internal instance memory is in use
8080	The identifier entered for the communications port is invalid
8081	Timeout, module error, internal error
8085	Error specifying the length at the LENGHT parameter. The specified length is "0" or greater than the maximum permitted value.
8090	Message length invalid, module invalid, message invalid
8091	Incorrect version in parameterization message
8092	Invalid record length in parameterization message

USS

Overview of USS instructions

Introduction

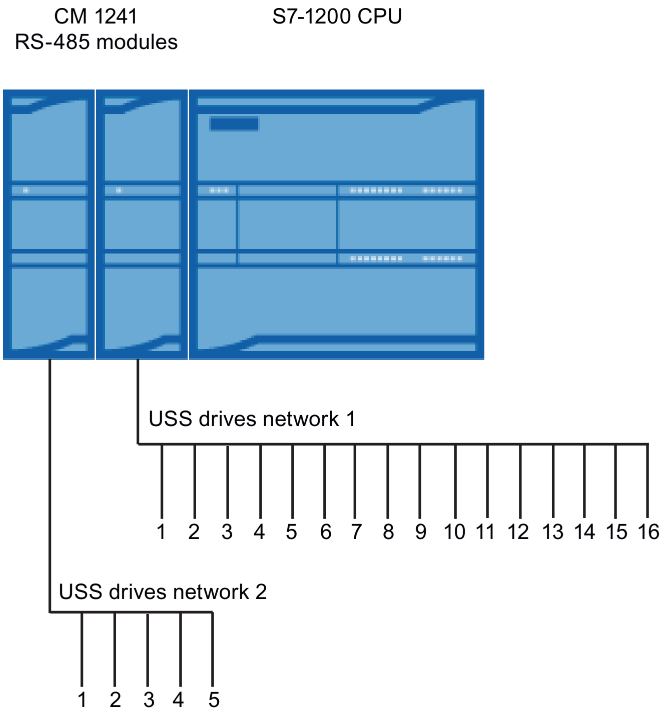
The USS instructions control the operation of drives that support the universal serial interface (USS). With the USS instructions, you can communicate with more than one drive via an RS-485 connection.

To do this, you require a CM 1241 RS-485 communications module or a CB 1241 RS-485 communications board. Up to three CM 1241 RS-485 modules and one CB 1241 RS-485 board can be installed in an S7-1200 CPU.

Each RS-485 port can operate up to sixteen drives.

The USS protocol uses a master/slave network for communication via a serial bus. The master uses an address parameter to send a message to a selected slave. A slave itself can never send without previously receiving a request. Direct exchange of messages between slaves is not possible. USS communication works in half duplex mode.

The following figure shows an example of a USS network diagram:



Requirements for using the USS protocol

General requirements for setting up a drive

- The use of 4 PKW words must be set up for the drives.
- The drives can be configured for 2, 4, 6 or 8 PZD words.
- The number of PZD words in the drive must match the PZD_LEN input of the "USS_DRV (Page 1919)" instruction of the drive.
- The baud rate of all drives must match the baud rate of the BAUD input parameter of the "USS_PORT (Page 1918)" instruction.
- The drive must be set up for remote control.
- USS must be specified for the desired frequency value on the COM connection of the drive.
- 1 to 16 must be set as the drive address. This address must match the address of the DRIVE input parameter of the "USS_DRV (Page 1919)" instruction.
- To control the direction of the drive, the polarity of the drive desired value must be set up.
- The RS-485 network must be connected correctly.

Definition: PKW / PZD area

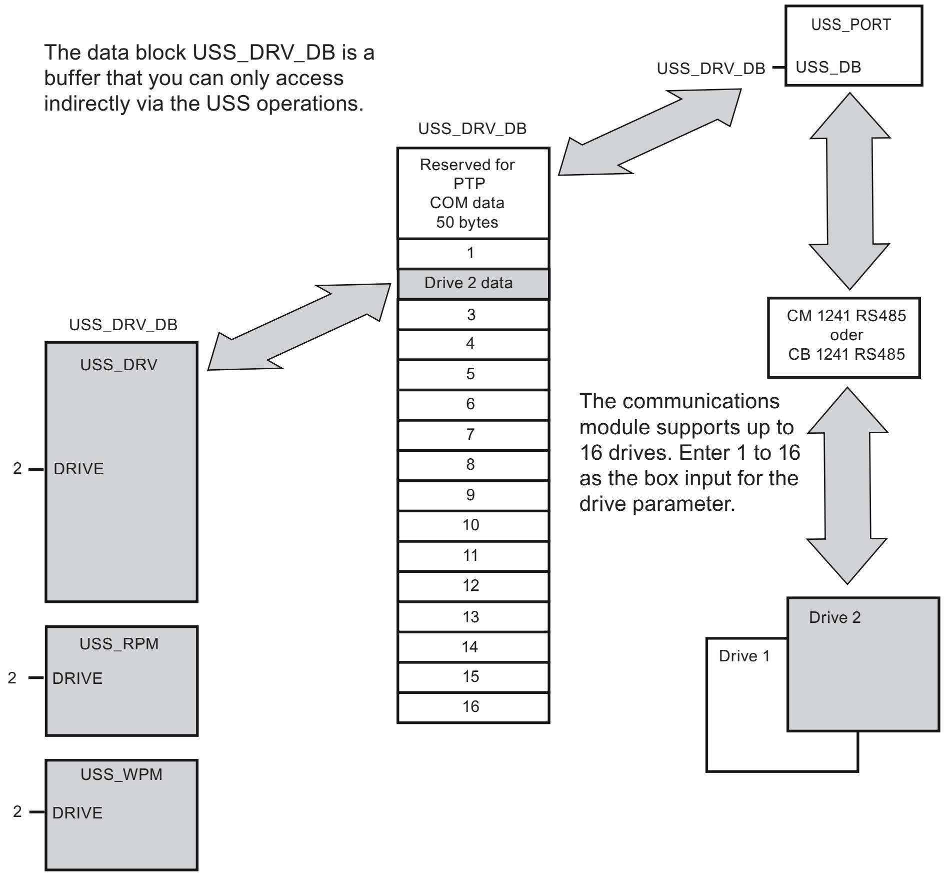
- The PKW area relates to the handling of the parameter-identifier-value interface.

  The PKW interface is not a physical interface but describes a mechanism that controls the exchange of parameters between two communications partners, in other words, reading and writing parameter values, parameter descriptions and corresponding texts and the handling of parameter changes due to spontaneous messages. All the tasks handled via the PKW interface are essentially tasks for operator control and monitoring, service and diagnostics.
- The PZD area includes the signals required for automation:
  - Control word(s), and setpoint(s) from the master to the slave
  - Status word(s), and actual value(s) from the slave to the master.

Both areas together result in the user data field. This is transferred by the master to the slave as a job frame or by the slave to the master as a reply frame.

Description

Each communication module CM 1241 RS485 supports a maximum of 16 drives. A single instance data block contains temporary memory and buffer functions for all drives in the USS network that are connected to a PtP communications module you installed. The USS instructions for these drives have shared access to the information in this data block.


The data block USS_DRV_DB is a buffer that you can only access indirectly via the USS operations.
USS_DRV_DB
USS_PORT
USS_DB
USS_DRV_DB
Reserved for PTP COM data 50 bytes
1
Drive 2 data
3
4
5
6
7
8
9
10
11
12
13
14
15
16
USS_DRV_DB
USS_DRV
2
DRIVE
USS_RPM
2
DRIVE
USS_WPM
2
DRIVE
CM 1241 RS485 oder CB 1241 RS485
The communications module supports up to 16 drives. Enter 1 to 16 as the box input for the drive parameter.
Drive 2
Drive 1

- All drives (max. 16) that are connected to an RS485 port are part of the same USS network. All drives that are connected to a different RS485 port are part of a different USS network. As the S7-1200 supports up to three CM 1241 RS485 modules, you can set up up to three USS networks, each having a maximum of 16 drives, thus a total of 48 USS drives are supported.
- Each USS network is managed via a unique data block (three data blocks are required for three USS networks with three CM 1241 RS485 modules). All instructions that belong to a USS network must share this data block. This includes all "USS_DRV (Page 1919)", "USS_PORT (Page 1918)", "USS_RPM (Page 1922)", and "USS_WPM (Page 1923)" instructions for controlling all drives in a USS network.

- The instruction "USS_DRV (Page 1919)" is a function block (FB). When you insert the instruction "USS_DRV" in the editor, in the "Call options" dialog you will be asked to assign a DB to the instruction.
  - If this is the first "USS_DRV" instruction in this program for this USS network, you can accept the default DB assignment (or, if necessary, change the name) and the new DB is created.
  - If this is not the first USS_DRV instruction for this network, you must select the appropriate DB that was previously assigned to this USS network in the drop-down list in the "Call options" dialog.
- All "USS_PORT (Page 1918), USS_RPM (Page 1922)", and "USS_WPM (Page 1923)" instructions are functions (FCs). When you insert these FCs in the editor, no DB is assigned. Instead, you have to assign the DB in question to the USS_DB input of these instructions (double-click on the parameter field and then on the icon to display the available DBs).
- The instruction "USS_PORT (Page 1918)" controls communication between the CPU and the drives via the PtP communications module. Whenever this instruction is called, communication with a drive is processed. Your program must call this function quickly enough so that the drives do not report a timeout. The instruction can be called from the main program or any interrupt OB.
- The function block "USS_DRV (Page 1919)" gives your program access to a specified drive in the USS network. Its inputs and outputs correspond to the states and operating functions of the drive. If there are 16 drives in the network, "USS_DRV" must be called at least 16 times in your program, i.e., once for each drive. How quickly these blocks are called depends on the required speed for controlling drive functions.

  You can only call the instruction "USS_DRV" from the main program OB.

⚠CAUTION
Call "USS_DRV", "USS_RPM", "USS_WPM" only from the main program OB. The instruction "USS_PORT" can be called from any OB, it is usually called from a time-delay interrupt OB. If the instruction "USS_PORT" is interrupted during execution, this may result in unexpected errors.

The "USS_RPM" and "USS_WPM" instructions are used to read and write the operating parameters of the drive. These parameters control the internal mode of operation of the drive. A definition of these parameters can be found in the drive manual.

Your program may also contain any number of these instructions; however only one read or write request can be active for a drive. You may only call the instructions "USS_RPM" and "USS_WPM" from the main program OB.

Calculating the time for communication with the drive

Communication with the drive is runs asynchronous to the cycle of the S7-1200. The S7-1200 runs through several cycles before communication with a drive is completed.

The interval of "USS_PORT" is the time required for a drive transaction. The following table shows the minimum intervals for "USS_PORT" for each baud rate. Calling the "USS_PORT" more frequently than the "USS_PORT" interval will not increase the number of transactions. The timeout interval of the drive is the period of time available to a transaction if 3 attempts are required to complete the transaction due to communications errors. By default, up to 2 further attempts are made for each transaction with the USS protocol.

Baud rate	Calculated minimum interval for calling USS_PORT (ms)	Drive message interval timeout per drive (ms)
1200	790	2370
2400	405	1215
4800	212.5	638
9600	116.3	349
19200	68.2	205
38400	44.1	133
57600	36.1	109
115200	28.1	85

USS_PORT: Edit communication via USS network

Description

The "USS_PORT" instruction handles communication over the USS network. In the program, use one "USS_PORT" instruction per PtP communications port to control the transmission to or from one drive.

All USS instructions that are assigned to one USS network and one PtP communications port must use the same instance data block.

Call

Your program must execute the "USS_PORT" instruction often enough to prevent timeouts in the drive. You should therefore call the "USS_PORT" instruction from a cyclic interrupt OB to prevent drive timeouts and keep the most recent USS data updates available for "USS_DRV (Page 1919)" calls.

Parameters

The following table shows the parameters of the "USS_PORT" instruction:

Parameter	Declaration	Data type	Memory area	Description
PORT	Input	PORT	D, L or constant	PtP communications port identifier Constant that can be referenced within the "Constants" tab of the default tag table.
BAUD	Input	DINT	I, Q, M, D, L or constant	Baud rate for USS communication.
USS_DB	Input	DINT	D	Reference to the instance DB of the "USS_DRV (Page 1919)" instruction.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR is set to TRUE if an error occurs. A corresponding error code will be output at the STATUS output.
STATUS (Page 1925)	Output	WORD	I, Q, M, D, L	Status value of the request. It indicates the result of the cycle or initialization. Additional information is available in the "USS_Extended_Error (Page 1925)" tag for some status codes.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

USS_DRV: Swap data with drive

Description

The "USS_DRV" instruction exchanges data with the drive by creating request messages and interpreting the drive response messages. A separate instruction must be used for each drive, but all USS instructions assigned to one USS network and one PtP communications module must use the same instance data block. You must create the DB name when you place the first "USS_DRV" instruction. Then reuse this DB that was created when the initial instruction was inserted.

When the "USS_DRV" instruction is executed the first time, the drive indicated by the USS address (parameter DRIVE) is initialized in the instance DB. After this initialization, subsequent "USS_PORT (Page 1918)" instructions can start communication with the drive at this drive number.

Changing the drive number requires a PLC STOP to RUN mode transition that initializes the instance DB. Input parameters are configured in the USS send buffer, and outputs are read from a "previous" valid response buffer if any exists. There is no data transmission during execution of the "USS_DRV" instruction. Communication with the drives takes place when "USS_PORT (Page 1918)" is executed. "USS_DRV" only configures the messages to be sent and interprets data that have been received from a previous request.

You can control the drive direction of rotation using either the DIR (BOOL) input or using the sign (positive or negative) at the SPEED_SP (REAL) input. The following table explains how these inputs work together to determine the drive direction, assuming the motor is wired for forward rotation.

SPEED_SP	DIR	Direction of rotation of drive
Value > 0	0	Reverse
Value > 0	1	Forward
Value < 0	0	Forward
Value < 0	1	Reverse

Parameters

Expand the box to display all the parameters by clicking the bottom of the box. The parameter connections that are grayed are optional and do not need to be assigned.

The following table shows the parameters of the "USS_DRV" instruction:

Parameter	Declaration	Data type	Memory area	Description
RUN	Input	BOOL	I, Q, M, D, L or constant	Drive start bit: If this parameter has the value TRUE, this input enables the drive to run at the preset speed.
OFF2	Input	BOOL	I, Q, M, D, L or constant	"Electrical stop" bit: If this parameter has the value FALSE, this bit cause the drive to coast to a stop without braking.
OFF3	Input	BOOL	I, Q, M, D, L or constant	Fast stop bit - If this parameter has the value FALSE, this bit causes a fast stop by braking the drive.
F_ACK	Input	BOOL	I, Q, M, D, L or constant	Fault acknowledge bit - This bit resets the fault bit on a drive. This bit is set after the fault is cleared to indicate to the drive that it no longer needs to indicate the previous fault.
DIR	Input	BOOL	I, Q, M, D, L or constant	Drive direction control - This bit is set to indicate that the direction is forward (when SPEED_SP is positive).
DRIVE	Input	USINT	I, Q, M, D, L or constant	Drive address: This input is the address of the USS drive. The valid range is drive 1 to drive 16.
PZD_LEN	Input	USINT	I, Q, M, D, L or constant	Word length - This is the number of words of PZD data. Valid values are 2, 4, 6, or 8 words. The default is 2.
SPEED_SP	Input	REAL	I, Q, M, D, L or constant	Speed setpoint - This is the speed of the drive as a percentage of configured frequency. A positive value specifies forward direction (when DIR has the value TRUE).
CTRL3	Input	WORD	I, Q, M, D, L or constant	Control word 3 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
CTRL4	Input	WORD	I, Q, M, D, L or constant	Control word 4 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
CTRL5	Input	WORD	I, Q, M, D, L or constant	Control word 5 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter

Parameter	Declaration	Data type	Memory area	Description
CTRL6	Input	WORD	I, Q, M, D, L or constant	Control word 6 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive.
CTRL7	Input	WORD	I, Q, M, D, L or constant	Control word 7 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
CTRL8	Input	WORD	I, Q, M, D, L or constant	Control word 8 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
NDR	Output	BOOL	I, Q, M, D, L	New data ready - If this parameter has the value TRUE, the bit indicates that the output contains data from a new communication request.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred - If this parameter has the value TRUE, this indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported on the ERROR and STATUS outputs of the "USS_PORT" instruction.
STATUS (Page 1925)	Output	WORD	I, Q, M, D, L	Status value of the request. It indicates the result of the cycle. This is not a status word returned from the drive.
RUN_EN	Output	BOOL	I, Q, M, D, L	Run enabled - This bit indicates whether the drive is running.
D_DIR	Output	BOOL	I, Q, M, D, L	Drive direction - This bit indicates whether the drive is running forward.
INHIBIT	Output	BOOL	I, Q, M, D, L	Drive inhibited - This bit indicates the state of the inhibit bit on the drive.
FAULT	Output	BOOL	I, Q, M, D, L	Drive fault - This bit indicates that the drive has registered a fault. The user must eliminate the fault and then set the F_ACK bit to clear this bit.
SPEED	Output	REAL	I, Q, M, D, L	Drive current speed (scaled value of drive status word 2) - The value of the speed of the drive as a percentage of configured speed.
STATUS1	Output	WORD	I, Q, M, D, L	Drive status word 1 - This value contains fixed status bits of a drive.
STATUS3	Output	WORD	I, Q, M, D, L	Drive status word 3 - This value contains a user-configurable status word on the drive.
STATUS4	Output	WORD	I, Q, M, D, L	Drive status word 4 - This value contains a user-configurable status word on the drive.
STATUS5	Output	WORD	I, Q, M, D, L	Drive status word 5 - This value contains a user-configurable status word on the drive.
STATUS6	Output	WORD	I, Q, M, D, L	Drive status word 6 - This value contains a user-configurable status word on the drive.
STATUS7	Output	WORD	I, Q, M, D, L	Drive status word 7 - This value contains a user-configurable status word on the drive.
STATUS8	Output	WORD	I, Q, M, D, L	Drive status word 8 - This value contains a user-configurable status word on the drive.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

USS_RPM: Readout parameters from the drive

Description

The "USS_RPM" instruction reads a parameter from the drive. All USS functions that are assigned to one USS network and one PtP communications module must use the same instance data block. "USS_RPM" must be called from the main program OB.

Parameters

The following table shows the parameters of the "USS_RPM" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Send request: If this parameter has the value TRUE, it indicates that a new read request is desired. This is ignored if the request for this parameter is already pending.
DRIVE	Input	USINT	I, Q, M, D, L or constant	Drive address: This input is the address of the USS drive. The valid range is drive 1 to drive 16.
PARAM	Input	UINT	I, Q, M, D, L or constant	Parameter number: This input specifies which drive parameter is written. The range of this parameter is 0 to 2047. See your drive manual for details on how to access any parameters above this range.
INDEX	Input	UINT	I, Q, M, D, L or constant	Parameter index: This input specifies which drive parameter index is to be written. This is a 16-bit value where the least significant byte is the actual index value with a range of (0 to 255). The most significant byte may also be used by the drive and is drive-specific. See your drive manual for additional information.
USS_DB	Input	VARIANT	D	Reference to the instance DB that is created and initialized when a "USS_DRV" instruction is inserted in your program.
DONE	Output	BOOL	I, Q, M, D, L	If this parameter has the value TRUE, it indicates that the VALUE output holds the previously requested read parameter value. This bit is set when the "USS_DRV" instruction recognizes the read response from the drive. This bit is reset when either: • you request the response data via another "USS_RPM" poll or • the second of the next two calls of "USS_DRV (Page 1919)" is executed

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred - If this parameter has the value TRUE, this indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported at the ERROR and STATUS outputs of the "USS_PORT (Page 1918)" instruction.
STATUS (Page 1925)	Output	WORD	I, Q, M, D, L	This is the status value of the request. It indicates the result of the read request. Additional information is available in the "USS_Extended_Error (Page 1925)" tag for some status codes.
VALUE	Output	WORD, INT, UINT, DWORD, DINT, UDINT, REAL	I, Q, M, D, L	This is the value of the parameter that was read and is valid only when the DONE bit has the value TRUE.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

USS_WPM: Change parameters in the drive

Description

The "USS_WPM" instruction modifies a parameter in the drive. All USS functions that are assigned to one USS network and one PtP communications module must use the same instance data block. "USS_WPM" must be called from the main program OB.

Note

EEPROM write operations

Beware of overusing the EEPROM write operation. Minimize the number of EEPROM write operations to extend the EEPROM life.

Parameters

The following table shows the parameters of the "USS_WPM" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Send request: If this parameter has the value TRUE, it indicates that a new write request is desired. This is ignored if the request for this parameter is already pending.
DRIVE	Input	USINT	I, Q, M, D, L or constant	Drive address: This input is the address of the USS drive. The valid range is drive 1 to drive 16.

Parameter	Declaration	Data type	Memory area	Description
PARAM	Input	UINT	I, Q, M, D, L or constant	Parameter number: This input specifies which drive parameter is written. The range of this parameter is 0 to 2047. See your drive manual for details on how to access any parameters above this range.
INDEX	Input	UINT	I, Q, M, D, L or constant	Parameter index: This input specifies which drive parameter index is to be written. This is a 16-bit value where the least significant byte is the actual index value with a range of (0 to 255). The most significant byte may also be used by the drive and is drive-specific. See your drive manual for additional information.
EEPROM	Input	BOOL	I, Q, M, D, L or constant	Store to drive EEPROM: If this parameter has the value TRUE, values written to the drive parameter will be stored in the drive EEPROM. If this parameter has the value FALSE, the value written is only temporarily saved and will be lost the next time the drive is switched on.
VALUE	Input	WORD, INT, UINT, DWORD, DINT, UDINT, REAL	I, Q, M, D, L or constant	The value of the parameter that is to be written. It must be valid on the transition of REQ.
USS_DB	InOut	VARIANT	D	This is a reference to the instance DB that is created and initialized when a "USS_DRV (Page 1919)" instruction is inserted in your program.
DONE	Output	BOOL	I, Q, M, D, L	If this parameter has the value TRUE, the VALUE input was written to the drive. This bit is set when the "USS_DRV (Page 1919)" instruction recognizes the write response from the drive. This bit is reset when either: You request the drive's confirmation that the write operation is complete via another "USS_WPM" poll or when the second of the next two calls to "USS_DRV (Page 1919)" is executed.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred: If this parameter has the value TRUE, this indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported at the ERROR and STATUS outputs of the "USS_PORT (Page 1918)" instruction.
STATUS (Page 1925)	Output	WORD	I, Q, M, D, L	This is the status value of the request. It indicates the result of the write request. Additional information is available in the "USS_Extended_Error (Page 1925)" tag for some status codes.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

Parameter STATUS of USS instructions

STATUS parameter

The following table contains the status codes of the USS operation that are output at the STATUS output of the USS instructions:

STATUS (W#16#....)	Description
0000	No error
8180	The length of the drive response did not match the characters received from the drive. The drive number where the error occurred is returned in the "USS_Extended_Error" tag. See the extended error description below this table.
8181	The parameter VALUE is not of the data type WORD, REAL, or DWORD
8182	User supplied a parameter value of the type word and received a DWORD or REAL from the drive in the response
8183	User supplied a parameter value of the type DWORD or REAL and received a word from the drive in the response
8184	Response telegram from drive had a bad checksum. The drive number where the error occurred is returned in the "USS_Extended_Error" tag. See the extended error description below this table.
8185	Illegal drive address (valid drive address range: 1-16)
8186	Speed set point out of valid range (valid speed SP range: -200% to 200%)
8187	Wrong drive number responded to the request sent. The drive number where the error occurred is returned in the "USS_Extended_Error" tag. See the extended error description below this table.
8188	Illegal PZD word length specified (valid range = 2, 4, 6 or 8 words)
8189	Illegal baud rate was specified
818A	Parameter request channel is in use by another request for this drive
818B	Drive has not responded to requests and retries. The drive number where the error occurred is returned in the "USS_Extended_Error" tag. See the extended error description below this table.
818C	Drive returned an extended error on a parameter request operation. See the extended error description below this table.
818D	Drive returned an illegal access error on a parameter request operation. See your drive manual for information of why parameter access may be limited
818E	Drive has not been initialized: This error code is returned to "USS_RPM (Page 1922)" or "USS_WPM (Page 1923)" when the "USS_DRV (Page 1919)" instruction for this drive has not been called at least once. This keeps the initialization of the first cycle of "USS_DRV (Page 1919)" from overwriting a pending parameter read or write request since it initializes the drive as a new entry. To eliminate this error, call the "USS_DRV (Page 1919) instruction" for this drive.
80Ax-80Fx	Specific errors returned from PtP (Point-to-Point) communication instructions called by the USS library: These error code values are not modified by the USS library and are defined in the PtP instruction descriptions.

USS_Extended_Error - USS drive extended error codes

USS drives support read and write access to a drive’s internal parameters. This feature allows distributed control and configuration of the drive. Drive parameter access operations can fail due to errors such as values out of range or illegal requests in a drive’s current mode. The drive generates an error code value that is returned in the "USS_Extended_Error" variable in the instance DB of the "USS_DRV (Page 1919)" instruction. This error code value is only valid for the last execution of the "USS_RPM (Page 1922)" or "USS_WPM (Page 1923)" instruction. The drive error code is put into the "USS_Extended_Error" tag when the value of STATUS is hexadecimal 818C. The error code of "USS_Extended_Error" depends on the drive variant. See the drive's manual for a description of the extended error codes for read and write parameter operations.

MODBUS

MB_COMM_LOAD: Configure port on the PtP module for Modbus RTU

Description

The "MB_COMM_LOAD" instruction configures a port for communication using the Modbus RTU protocol. The following hardware can be used for this:

- Up to three point-to-point modules (PtP) CM 1241 RS485 or CM 1241 RS232
- A communications board CB 1241 RS485 in addition to this

After configuration of the port, you communicate over Modbus by executing the "MB_SLAVE" or "MB_MASTER" instruction.

Call

"MB_COMM_LOAD" must be called once to configure the port for the Modbus RTU protocol. On completion of the configuration, the port can be used by the "MB_MASTER (Page 1929)" and "MB_SLAVE (Page 1936)" instructions.

"MB_COMM_LOAD" only needs to be called again if one of the communication parameters has to be modified. Each "MB_COMM_LOAD" call deletes the communications buffer. To avoid data loss during communication, you should not call the instruction unnecessarily.

One instance of "MB_COMM_LOAD" must be used to configure the port of each communication module that is used for Modbus communication. You assign a unique "MB_COMM_LOAD" instance data block for each port that you use. The S7-1200 CPU is limited to three communication modules.

An instance data block is assigned when you insert the "MB_MASTER (Page 1929)" or "MB_SLAVE (Page 1936)" instruction. This instance data block is referenced when you specify the MB_DB parameter on the "MB_COMM_LOAD" instruction.

Parameters

The following table shows the parameters of the "MB_COMM_LOAD" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Execution of the instruction on a rising edge.
PORT	Input	UINT	I, Q, M, D, L or constant	ID of the communications port: After you have inserted the communications module in the device configuration, the port ID appears in the drop-down list at the PORT box connection. This constant can also be referenced within the "Constants" tab of the tag table.
BAUD	Input	UDINT	I, Q, M, D, L or constant	Baud rate selection: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200 All other values are invalid.
PARITY	Input	UINT	I, Q, M, D, L or constant	Parity selection: • 0 – None • 1 – Odd • 2 – Even
FLOW_CTRL	Input	UINT	I, Q, M, D, L or constant	Flow control selection: • 0 – (default) No flow control • 1 – Hardware flow control with RTS always ON (does not apply to RS485 ports) • 2 - Hardware flow control with RTS switched
RTS_ON_DLY	Input	UINT	I, Q, M, D, L or constant	RTS on-delay selection: • 0 – (default) No delay of RTS active until the first character of the message is transmitted. • 1 to 65535 – Delay in milliseconds of "RTS active" until the first character of the message is transmitted (does not apply to RS-485 ports). RTS delays must be applied independent of the FLOW_CTRL selection.
RTS_OFF_DLY	Input	UINT	I, Q, M, D, L or constant	RTS off-delay selection: • 0 – (default) No delay after the last character transmitted until "RTS inactive" • 1 to 65535 – Delay in milliseconds after the last character transmitted until "RTS inactive" (does not apply to RS-485 ports). RTS delays must be applied independent of the FLOW_CTRL selection.

Parameter	Declaration	Data type	Memory area	Description
RESP_TO	Input	UINT	I, Q, M, D, L or constant	Response timeout: Time in milliseconds allowed by "MB_MASTER (Page 1929)" for the slave to respond. If the slave does not respond in this time, "MB_MASTER (Page 1929)" repeats the request or terminates the request with an error if the specified number of retries has been sent. 5 ms to 65535 ms (default = 1000 ms).
MB_DB	Input	VARIANT	D	A reference to the instance data block of the "MB_MASTER (Page 1929)" or "MB_SLAVE (Page 1936)" instructions. After you insert "MB_SLAVE (Page 1936)" or "MB_MASTER (Page 1929)" in your program, the DB identifier appears in the drop-down list at the MB_DB box connection.
DONE	Output	BOOL	I, Q, M, D, L	Execution of instruction completed without error.
ERROR	Output	BOOL	I, Q, M, D, L	Error: • 0 – No error detected • 1 – Indicates that an error was detected. An error code is output in the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Port configuration error code

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

STATUS parameter

Error code(W#16#...)	Description
0000	No error
8180	Invalid value for the port ID (wrong address for the communications module).
8181	Invalid baud rate value.
8182	Invalid parity value.
8183	Invalid flow control value.
8184	Invalid value for the timeout of the response (the time before which a timeout is reported must be at least 25 ms).
8185	Incorrect pointer in the MB_DB parameter to the instance DB of the "MB_MASTER (Page 1929)" or "MB_SLAVE (Page 1936)" instruction.

MB_MASTER: Communicate via the PtP port as Modbus master

Description of MB_MASTER

Description

The "MB_MASTER" instruction allows your program to communicate as a Modbus master using a port on a point-to-point module (CM) or a communications board (CB). You can access data in one or more Modbus slave devices.

Before the "MB_MASTER" instruction can communicate with a port, "MB_COMM_LOAD (Page 1926)" must first execute.

An instance DB is created when you insert the "MB_MASTER" instruction in your program. You specify this instance DB in the MB_DB input parameter of the "MB_COMM_LOAD (Page 1926)" instruction.

Rules for Modbus master communication

- A port used for Modbus master requests cannot be used for "MB_SLAVE".
- A port can be used for one or more "MB_MASTER" calls if the same instance DB is used.
- The Modbus instructions do not use communication interrupt events to control the communication process. Your program must poll the "MB_MASTER" instruction for completed send and receive operations.
- Calling the instruction:
  - Call the "MB_MASTER" instruction if possible in a cyclic program OB. The instruction can only be called in a time delay or cyclic interrupt OB.
  - Do not call more than one "MB_MASTER" instruction in organization blocks with different priority classes. If a "MB_MASTER" instruction executes "preemptively" from a higher priority class, the instruction may execute incorrectly.
  - Do not call the "MB_MASTER" instruction in a startup, diagnostics or time error OB.
- After a transfer has started, the EN parameter (LAD/FBD) must remain set to the value "1" until the DONE or ERROR output parameter is set to "1" by the instruction. A renewed call by the REQ parameter while the instruction is executing causes an error. After the instruction executes, the bit in the REQ parameter remains set for the time specified by the BLOCKED_PROC_TIMEOUT parameter in the instance DB.
- If "MB_MASTER" sends a request to a slave, make sure that "MB_MASTER" continues to execute until the response from the slave arrives.

Parameters

The following table shows the parameters of the "MB_MASTER" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ (Page 1932)	Input	BOOL	I, Q, M, D, L	Request input: • 0 – No request • 1 – Request to transmit data to Modbus slave(s)
MB_ADDR	Input	UINT	I, Q, M, D, L or constant	Modbus RTU station address: • Default address range: 0 to 247 • Extended address range: 0 to 65535 The value "0" is reserved for broadcasting a message to all Modbus slaves. Modbus function codes 05, 06, 15, and 16 are the only function codes supported for broadcast.
MODE (Page 1932)	Input	USINT	I, Q, M, D, L or constant	Mode selection: Specifies the type of request: Read, write, or diagnostics: Refer to the Modbus functions table for details.
DATA_ADDR (Page 1932)	Input	UDINT	I, Q, M, D, L or constant	Starting address in the slave: Specifies the starting address of the data to be accessed in the Modbus slave. You will find the valid addresses in the Modbus functions table.
DATA_LEN	Input	UINT	I, Q, M, D, L or constant	Data length: Specifies the number of bits or words to be accessed in this request. You will find the valid lengths in the Modbus functions table.
DATA_PTR (Page 1933)	Input	VARIANT	M, D	Points to the DB or bit memory address of the CPU for the data to be written or read. For a DB, this must be created with the "Standard - compatible with S7-300/400" access type.
DONE	Output	BOOL	I, Q, M, D, L	• 0: Transaction not completed • 1: Transaction completed without error
BUSY	Output	BOOL	I, Q, M, D, L	• 0: No "MB_MASTER" transaction in progress • 1: "MB_MASTER" transaction in progress
ERROR	Output	BOOL	I, Q, M, D, L	• 0: No error • 1: Error, the error code is indicated by the STATUS parameter
STATUS	Output	WORD	I, Q, M, D, L	Execution condition code

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

STATUS parameter

Table 9- 56 Communications and configuration error messages of the instruction

Error code (W#16#....)	Description
0000	No error
80C8	Slave timeout. Check the baud rate, parity and the connectors on the slave.
80D1	The receiver issued a flow control request to suspend an active transmission and never re-enabled the transmission within the wait time. This error is also generated during hardware flow control if the recipient does not detect CTS within the wait time.
80D2	The send request was aborted because no DSR signal is received from the DCE.
80E0	The message was terminated because the receive buffer is full.
80E1	The message was terminated as a result of a parity error.
80E2	The message was terminated as a result of a framing error.
80E3	The message was terminated as a result of an overrun error.
80E4	The message was terminated as a result of the specified length exceeding the total buffer size.
8180	Invalid value for the port ID.
8186	Invalid Modbus station address
8188	The MODE parameter has an invalid value for a broadcast call.
8189	Invalid data address value.
818A	Invalid data length value.
818B	Invalid pointer to the local data source/destination: Size not correct
818C	The DATA_PTR parameter has an invalid pointer. Use a pointer to a bit memory area or a DB with the "Standard - compatible with S7-300/400" access type.
8200	Port is busy processing a send request

Table 9- 57 Error messages of the Modbus protocol

Error code (W#16#....)	Response code of slave	Description
8380	-	CRC error
8381	01	Function code not supported
8382	03	Data length error
8383	02	Error in the data address or address outside the valid range of DATA_PTR
8384	> 03	Data value error
8385	03	Data diagnostic code value not supported (function code 08)
8386	-	Function code of the response does not match the function code of the query.
8387	-	Response from wrong slave
8388	-	The response of the slave to a write call is not correct. The data sent by the slave does not match the query from the master.

Parameter REQ

Description

- REQ = FALSE: No request
- REQ = TRUE: Request to transmit data to Modbus slave(s)

You must supply this input through a positive edge-triggered contact on the first call of "MB_MASTER" execution. The edge-triggered pulse will invoke the transmission request once. All inputs are captured and kept unchanged for the duration of a request and response triggered by this input.

While an instance of the "MB_MASTER" executes, no further instance of the instruction can be called. If there is a further instance call by the REQ parameter while "MB_MASTER" is executing, no automatic follow-on call will be started. To be able to call the instance again, the instruction must first be completed before it can be called again by the REQ parameter.

DATA_ADDR and MODE parameters

Description

You specify the start address for data access to the Modbus slave using the DATA_ADDR parameter.

With the MODE parameter and the Modbus address, you specify the function code to be transferred to the Modbus slave. The following table shows the relationship between the MODE parameter, the function code and Modbus address range.

MODE	Modbus function	Data length	Operation and data	Modbus address
0	01	1 to 2000 1 to 1992 (1)	Read output bits: 1 to (1992 or 2000) bits per query	1 to 9999
0	02	1 to 2000 1 to 1992 (1)	Read input bits: 1 to (1992 or 2000) bits per query	10001 to 19999
0	03	1 to 125 1 to 124 (1)	Read holding register: 1 to (124 or 125) WORD per query	40001 to 49999 or 400001 to 465535
0	04	1 to 125 1 to 124 (1)	Read input WORD: 1 to (124 or 125) WORD per query	30001 to 39999
1	05	1	Writing an output bit: One bit per query	1 to 9999
1	06	1	Writing a holding register: 1 WORD per query	40001 to 49999 or 400001 to 465535
1	15	2 to 1968 2 to 1960 (1)	Writing multiple output bits: 2 to (1960 or 1968) bits per query	1 to 9999
1	16	2 to 123 2 to 122 (1)	Writing multiple holding registers: 2 to (122 or 123) WORD per query	40001 to 49999 or 400001 to 465535

MODE	Modbus function	Data length	Operation and data	Modbus address
2	15	1 to 1968 2 to 1960 [(1)]	Writing one or more output bits: 1 to (1960 or 1968) bits per query	1 to 9999
2	16	1 to 123 2 to 122 [(1)]	Writing one or more holding registers: 1 to (122 or 123) WORD per query	40001 to 49999 or 400001 to 465535
11	11	0	Reading out the communications status word of the slaves and the event counter: The status word indicates execution of the instruction (0: is not executing; 0xFFFF: is executing). The event counter is incremented each time a message is transferred successfully. The DATA_ADDR and DATA_LEN parameters of the "MB_MASTER" instruction are ignored with this function.	-
80	08	1	Checking the slave status by reading the error code (0x0000): 1 WORD per query	-
81	08	1	Resetting the event counter of the slave with the diagnostics code 0x000A: 1 WORD per query	-
3 to 10, 12 to 79, 82 to 2555			Reserved	-
[(1)] For the "Extended address range", the maximum data length is reduced by one byte or one WORD depending on which data type is used for the function.				

Parameter DATA_PTR

Description

The DATA_PTR parameter is a pointer to a data block or bit memory from which the data should be written or read. If you use a data block, create a global data block with the "Standard - compatible with S7-300/400" access type.

Data block structures for the DATA_PTR parameter

- These data types are valid for **reading of words** of Modbus addresses 30001 to 39999, 40001 to 49999, and 400001 to 465536 and also for **writing of words** to Modbus addresses 40001 to 49999 and 400001 to 465536.
  - Standard array of WORD, UINT, or INT data types (see below).
  - Named WORD, UINT, or INT structure where each element has a unique name and 16 bit data type.
  - Named complex structure where each element has a unique name and 16 bit or 32 bit data type.
- For reading and writing of bits of Modbus addresses 00001 to 09999 and 10001 to 19999.
  - Standard array of Boolean data types.
  - Named Boolean structure of uniquely named Boolean variables.
- Although not required, it is recommended that each "MB_MASTER" instruction has its own separate memory area in a global data block. The reason for this recommendation is that there is a greater possibility of data corruption if multiple "MB_MASTER" instructions are reading and writing the same area of a global data block.
- The memory areas for DATA_PTR do not need to be in the same global data block. You can create one data block with multiple areas for Modbus read operations, one data block for Modbus write operations, or one data block for each slave station.

Instance DB of the "MB_MASTER" instruction

Static variables of the instance DB

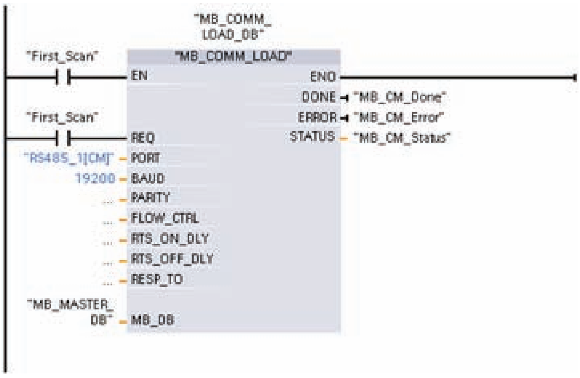
The following table describes the static variables of the instance DB of the instruction that you can use in the user program.

Variable	Data type	Description
MB_STATE	UINT	Internal status of the Modbus instruction
BLOCKED_ PROC_TIMEOUT	REAL	Time between completion of the instruction call and resetting the ACTIVE bit in the instance DB. The time buffer is used to avoid execution of the instruction being terminated before a job has been sent completely. The default time is 500 ms.
EXTENDED_ ADDRESSING	BOOL	Configuring addressing: • 0: Default address area (1 byte) • 1: Extended address area (2 bytes) For additional information, refer to the section EXTENDED_ADDRESSING: Instance DB of the "MB_SLAVE" instruction (Page 1940)

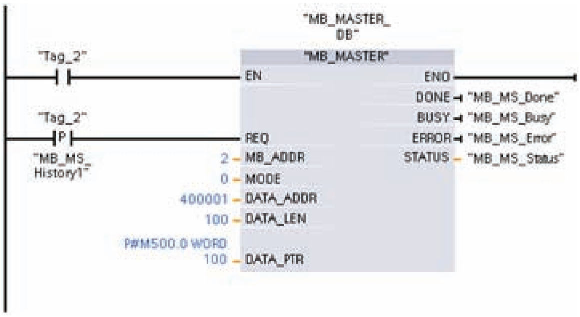
Sample program for a Modbus master

Networks (LAD)

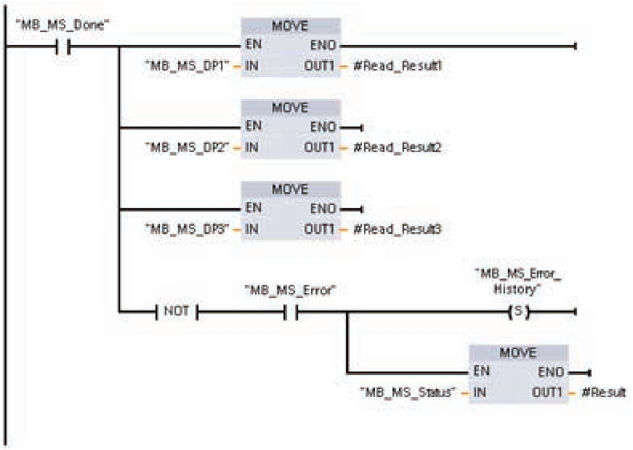
Network 1: Initialize parameters of the RS-485 module only once during the first cycle.



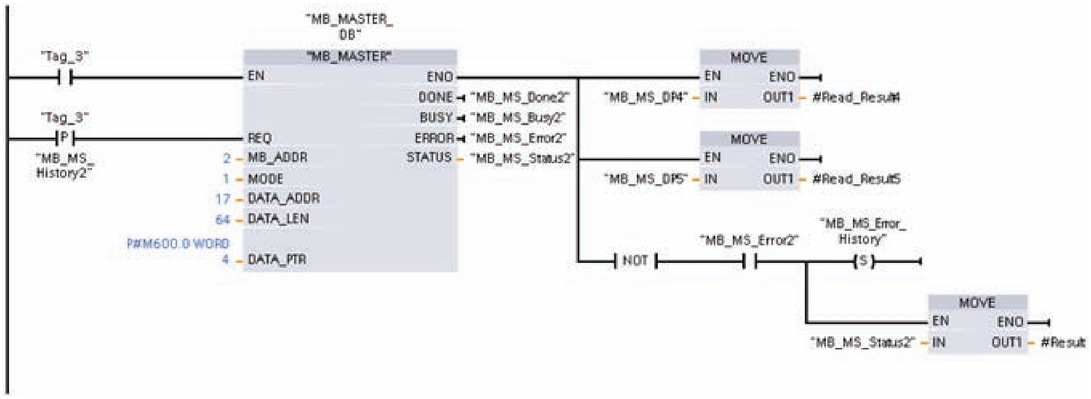
Network 2: Read 100 words from the holding register of the slave.



Network 3: This is an optional network that displays the values of the first 3 words as soon as the read operation has executed.



Network 4: Write 64 bits to the process image output, starting at slave address Q2.0.



MB_SLAVE: Communicate via the PtP port as Modbus slave

Description of MB_SLAVE

Description

The "MB_SLAVE" instruction allows your program to communicate as a Modbus slave using a port on a point-to-point module (PtP) or a communications board (CB). A Modbus RTU master can issue a request and then your program responds via "MB_SLAVE" execution.

You must assign a unique instance data block when you insert the "MB_SLAVE" instruction in your program. This instance data block is used when you specify it at the MB_DB parameter of the "MB_COMM_LOAD (Page 1926)" instruction.

Modbus communication function codes (1, 2, 4, 5, and 15) can read and write bits and words directly in the process image input and process image output in the target system. The following table shows the mapping of Modbus addresses to the process image in the CPU.

Modbus functions of "MB_SLAVE"						S7-1200	
Codes	Function	Data area	Address range			Data area	CPU address
01	Read bits	Output	1	to	8192	Process image output	Q0.0 to Q1023.7
02	Read bits	Input	10001	to	18192	Process image input	I0.0 to I1023.7
04	Read words	Input	30001	to	30512	Process image input	IW0 to IW1022
05	Write bit	Output	1	to	8192	Process image output	Q0.0 to Q1023.7
15	Write bits	Output	1	to	8192	Process image output	Q0.0 to Q1023.7

Modbus communication function codes (function codes 3, 6, 16) use a separate holding register. To do this, you can use bit memory or a data block with the "Standard - compatible with S7-300/400" access type.

You specify the type of the holding register using the MB_HOLD_REG parameter of the MB_SLAVE instruction. The following table shows the mapping of the Modbus holding register to the DB address of MB_HOLD_REG in the target system.

Modbus functions of "MB_SLAVE"				S7-1200	
Codes	Function	Data area	Address range (WORD number)	Address in the DB (BYTE number)	Bit memory address (BYTE number)
03	Read words	Holding register	40001 to 49999 or	DW0 to DW19998 or	MW0 to CPU limit
			400001 to 465535	DW0 to DW131068	
06	Write word	Holding register	40001 to 49999 or	DW0 to DW19998 or	
			400001 to 465535	DW0 to DW131068	
16	Write words	Holding register	40001 to 49999 or	DW0 to DW19998 or	
			400001 to 465535	DW0 to DW131068	

The table below shows the supported Modbus diagnostic functions.

S7-1200 "MB_SLAVE" Modbus diagnostic functions		
Codes	Subfunction	Description
08	0000H	Return query data echo test: The "MB_SLAVE" instruction returns the echo of a received data word to a Modbus master.
08	000AH	Clear communication event counter: The "MB_SLAVE" instruction clears the communication event counter that is used for Modbus function 11.
11	-	Get communication event counter: The "MB_SLAVE" instruction uses an internal communication event counter for recording the number of successful Modbus read and write requests that are sent to the Modbus slave. The counter is not incremented on any Function 8, Function 11, or broadcast requests. It is also not incremented on any requests that result in a communication error (for example, parity or CRC errors).

The "MB_SLAVE" instruction supports broadcast write requests from Modbus masters as long as the requests include access to valid addresses.

Regardless of the validity of a request, "MB_SLAVE" gives no response to a Modbus master as the result of a broadcast request.

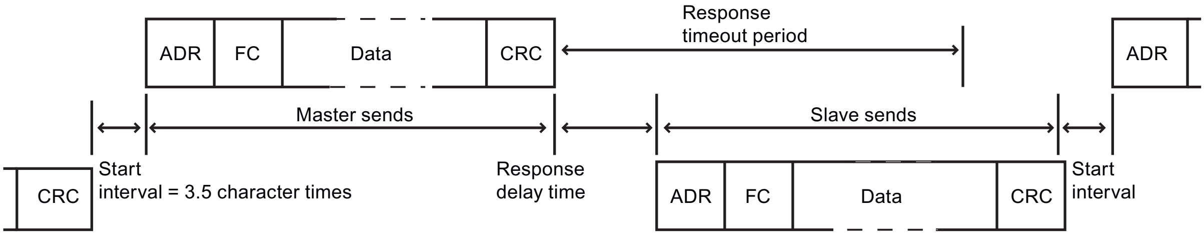
Rules of Modbus slave communication

- "MB_COMM_LOAD" must be executed to configure a port, before the "MB_SLAVE" instruction can communicate with this port.
- If a port is to respond as a slave to a Modbus master, then that port cannot be used by "MB_MASTER (Page 1929)". Only one instance of "MB_SLAVE" can be used with a given port.

- The Modbus instructions do not use communication interrupt events to control the communication process. Your program must control the communication process by polling the "MB_SLAVE" instruction for completed send and receive operations.
- The "MB_SLAVE" instruction must be executed periodically at a rate that allows it to make a timely response to incoming requests from a Modbus master. It is therefore advisable to call the instruction in a cyclic program OB. Calling the "MB_SLAVE" instruction in an interrupt OB is possible but not advisable since it can lead to long delays in execution.

Frequency of execution of "MB_SLAVE"

The "MB_SLAVE" instruction must be executed periodically to receive each request from the Modbus master and to respond as required. The frequency of execution of "MB_SLAVE" is dependent upon the specified response timeout period of the Modbus master. This is illustrated in the following diagram.



The response timeout period is the amount of time a Modbus master waits for the start of a response from a Modbus slave. This time period is not defined by the Modbus protocol, but rather by a parameter of each Modbus master. The frequency of execution (time between one execution and the next execution) of "MB_SLAVE" must be based on the particular parameters of your Modbus master. As a minimum, you should execute "MB_SLAVE" twice within the response timeout period of the Modbus master.

Parameters

The following table shows the parameters of the "MB_SLAVE" instruction:

Parameter	Declaration	Data type	Memory area	Description
MB_ADDR	Input	UINT	I, Q, M, D, L or constant	Station address of the Modbus slave • Default address range: 0 to 247 • Extended address range: 0 to 65535
MB_HOLD_REG	Input	VARIANT	D	Pointer to the Modbus holding register DB. The DB must be created with the "Standard - compatible with S7-300/400" access type.
NDR	Output	BOOL	I, Q, M, D, L	New data ready: • 0: No new data • 1: Indicates that new data has been written by the Modbus master

Parameter	Declaration	Data type	Memory area	Description
DR	Output	BOOL	I, Q, M, D, L	Read data: • 0: No data read • 1: Indicates that data has been read by the Modbus master
ERROR	Output	BOOL	I, Q, M, D, L	• 0: No error detected • 1: Error, a corresponding error code is output in the STATUS
STATUS	Output	WORD	I, Q, M, D, L	Error code

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

STATUS parameter

STATUS(W#16 #....)	Description	
80C8	The specified response timeout (refer to RCVTIME or MSGTIME) is "0".	
80D1	The receiver issued a flow control request to suspend an active transmission and never re-enabled the transmission within the wait time. This error is also generated during hardware flow control if the recipient does not detect CTS within the wait time.	
80D2	The transmit request was aborted because no DSR signal is received from the DCE.	
80E0	The message was terminated because the receive buffer is full	
80E1	The message was terminated as a result of a parity error	
80E2	The message was terminated as a result of a message frame error	
80E3	The message was terminated as a result of a overrun error	
80E4	The message was terminated as a result of the specified length exceeding the total buffer size	
8180	Invalid value for the port ID.	
8186	Invalid Modbus station address	
8187	Invalid pointer to MB_HOLD_REG-DB	
818C	Pointer to a type safe DB type MB_HOLD_REG (must be a Classic DB type)	
Response code sent to Modbus master (B#16#...)		
8380	No response	CRC error
8381	01	Function code not supported or not supported in a broadcast
8382	03	Data length error
8383	02	Error in the data address or address outside the valid range of MB_HOLD_REG
8384	03	Data value error
8385	03	Data diagnostic code value not supported (function code 08)

Instance DB of the "MB_SLAVE" instruction

Static variables of the instance DB

The following table describes the static variables of the instance DB of the instruction that you can use in the user program. Your program can write values to the HR_Start_Offset and Extended_Addressing variables and control the Modbus slave operations.

The other variables can be read to monitor the Modbus status.

Variable	Data type	Description
HR_Start_Offset	WORD	Start address of the Modbus holding register (default = " 0")
Extended_ Addressing	BOOL	Configuring addressing: • 0: Default address area (1 byte) • 1: Extended address area (2 bytes)
Request_Count	WORD	Total number of queries received by the slave
Slave_Message_ Count	WORD	Number of queries sent to this specific slave
Bad_CRC_Count	WORD	Number of received queries with CRC errors
Broadcast_Count	WORD	Number of received broadcast queries
Exception_Count	WORD	Number of Modbus-specific errors that require the return of an exception
Success_Count	WORD	The number of requests received for this specific slave without protocol errors

HR_Start_Offset

The addresses of the Modbus holding register start at 40001 or 400001. These addresses correspond to the start address of the holding register in the target system memory. Using the HR_Start_Offset variable, you can set the offset to a different start address.

Example: A holding register starts at MW 100 and has a length of 100 WORD. With an offset of 20 in the HR_Start_Offset parameter, the holding register begins at address 40021 instead of 40001. Each address below 40021 and above 400119 causes an addressing error.

	HR_Start_Offset = 0			HR_Start_Offset = 20	
	Modbus word address	S7-1200 byte address		Modbus word address	S7-1200 byte address
Minimum	40001	MW100		40021	MW100
Maximum	40099	MW198		40119	MW198

Extended_Addressing

To address the Modbus slave, a single byte (default address range) or a double byte (extended address range) can be configured. Extended addressing is used to address more than 247 devices in a single network. If you decide on extended addressing, you can address a maximum of 64,000 addresses. Below, you will see a frame of Modbus function 1 as an example.

Table 9- 58 Slave address with one byte (byte 0)

Function 1	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	
Request	Slave address	F code	Start address		Length of the coils		
Valid response	Slave address	F code	Length	Coil data			
Error response	Slave address	0x81	E code				

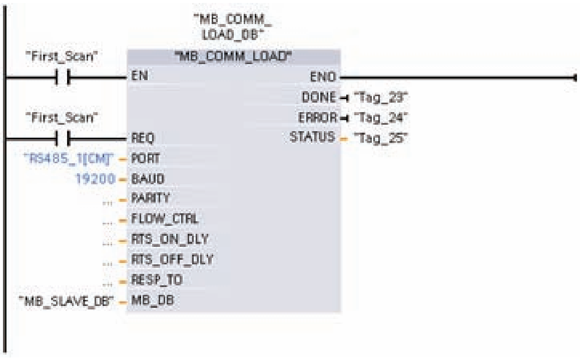
Table 9- 59 Slave address with two bytes (byte 0 and byte 1)

Function 1	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Request	Slave address		F code	Start address		Length of the coils	
Valid response	Slave address		F code	Length	Coil data		
Error response	Slave address		0x81	F code			

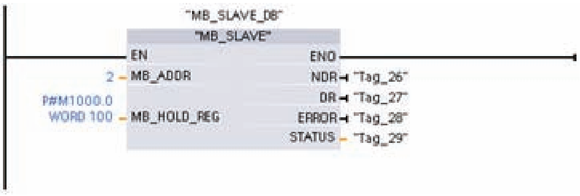
Sample program for a Modbus slave

Networks (LAD)

Network 1: Initialize parameters of the RS-485 module only once during the first cycle.



Network 2: Check the requests of the Modbus master in every cycle. 100 words starting at MW1000 are configured for the Modbus holding register.



MODBUS TCP

MB_CLIENT: Communicating via PROFINET as a Modbus TCP client

Description of MB_CLIENT

Description

The "MB_CLIENT" instruction communicates as a Modbus TCP client via the PROFINET connection of the S7-1200 CPU. To use the instruction, you do not require any additional hardware module. With the "MB_CLIENT" instruction, you establish a connection between the client and the server, send requests and receive responses and control connection termination of the Modbus TCP server.

Parameters

The following table shows the parameters of the "MB_CLIENT" instruction:

Parameters	Declaration	Data type	Description
REQ (Page 1945)	Input	BOOL	Communications request to the Modbus TCP server on a rising edge.
DISCONNECT (Page 1945)	Input	BOOL	With this parameter, you control the establishment and termination of the connection to the Modbus server: • 0: Establish a communications connection to the specified IP address and port number. • 1: Disconnect the communications connection. No other function is executed during connection termination.
CONNECT_ID	Input	WORD	Unique ID to identify the connection. Each instance of the instructions "MB_CLIENT" and "MB_SERVER (Page 1949)" must be assigned a unique connection ID.
IP_OCTET_1	Input	BYTE	1st Octet of the IP address* of the Modbus TCP server.
IP_OCTET_2	Input	BYTE	2nd Octet of the IP address* of the Modbus TCP server.
IP_OCTET_3	Input	BYTE	3rd Octet of the IP address* of the Modbus TCP server.
IP_OCTET_4	Input	BYTE	4th Octet of the IP address* of the Modbus TCP server.

Parameters	Declaration	Data type	Description
IP_PORT	Input	WORD	IP port number of the server to which the client establishes the connection and communicates using the TCP/IP protocol (default value: 502).
MB_MODE (Page 1945)	Input	USINT	Selects of the mode of the request (read, write or diagnostics).
MB_DATA_ADDR (Page 1945)	Input	UDINT	Start address of the data accessed by the "MB_CLIENT" instruction.
MB_DATA_LEN	Input	UINT	Data length: Number of bits or words for the data access (see MB_MODE and MB_DATA_ADDR parameters - data length).
MB_DATA_PTR (Page 1947)	InOut	VARIANT	Pointer to the Modbus data register: The register is a buffer for the data received from the Modbus server or to be sent to the Modbus server. The pointer must reference a data block or a memory area.
DONE	Out	BOOL	The bit at output parameter DONE is set to "1" as soon as the last job is completed without errors.
BUSY	Out	BOOL	• 0: No "MB_CLIENT " job in progress • 1: "MB_ CLIENT " job in progress
ERROR	Out	BOOL	• 0: No error • 1: Error occurred. The cause of error is indicated by the STATUS parameter.
STATUS (Page 1948)	Out	BOOL	Error code of the instruction.
* 8-bit long component of the 32-bit IPv4 IP address of the Modbus TCP server.			

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Note

Consistent input data during an "MB_CLIENT" call

When a Modbus client calls a Modbus instruction, the status of the input parameters is stored internally and then compared at the next call. The comparison is used to determine whether this particular call initialized the current request. Several "MB_CLIENT" calls can be executed if you use a common instance DB. The values of the input parameters must not be changed, as long as an "MB_CLIENT" is executing. If the input parameters are changed during execution, "MB_CLIENT" cannot be used to check whether or not the instance is currently executing.

Multiple client connections

A Modbus TCP client can support several TCP connections and the maximum number of connections depends on the CPU being used. The total number of connections of one CPU, including those of the Modbus TCP clients and server must not exceed the maximum number of supported connections. Modbus TCP connections can also be shared by client and/or server connections.

With individual client connections, remember the following rules:

- Each "MB_CLIENT" connection must use a unique instance DB.
- For each "MB_CLIENT" connection, a unique server IP address must be specified.
- Each "MB_CLIENT" connection requires a unique connection ID.

  The relevant individual connection ID must be used for each individual instance DB of the instruction. The connection ID and instance DB belong together in pairs and must be unique for each connection.
- Unique numbers for the IP Ports may or may not be required depending on the server configuration.

Static variables of the instruction

The following table describes the editable static variables of the instance data block of the "MB_CLIENT" instruction.

Variable	Data type	Start value	Description
Blocked_Proc_ Timeout	REAL	5.0	Wait time in seconds before the static variable ACTIVE is reset if there is a blocked Modbus instance. This can, for example, occur if a client request is output and the execution of the client function aborts before the request was fully executed. The maximum wait time is 55 seconds.
MB_Transaction_ID	WORD	1	Transaction ID of the Modbus TCP protocol. The start value of "1" should only be changed if the Modbus TCP server requires a different value.
MB_Unit_ID	WORD	65535	Unit ID of the Modbus protocol. The variable corresponds to the slave address of the Modbus RTU protocol. Change this value only if the Modbus TCP server can be used and as a gateway and is controlled by the application program within the Modbus server.
RCV_TIMEOUT	REAL	2.0	Time interval in seconds in which the "MB_CLIENT" instruction waits for a response from the server.

See also

MB_CLIENT example 1: Send several requests via a TCP connection (Page 1953)

MB_CLIENT example 2: Send multiple requests via several TCP connections (Page 1954)

MB_CLIENT example 3: Respond to request to write from the Modbus server (Page 1955)

MB_CLIENT example 4: Coordinate several requests (Page 1956)

REQ and DISCONNECT parameters

Description

If no instance of the "MB_CLIENT" instruction is executing and if the value of the DISCONNECT parameter is "0", a new job executes on a rising edge of the REQ parameter. If there is not yet a connection, this is established during execution.

If the same instance of the "MB_CLIENT" instruction executes again (DISCONNECT = 0 and REQ = 1), before the active job was executed, this is not executed on completion of the active job. A new job can only be started on completion of the active job (REQ = 1).

You can monitor the status of execution with the DONE parameter. You can use this to monitor the status of execution if the "MB_CLIENT" instruction is executed sequentially.

See also

Description of MB_CLIENT (Page 1942)

MB_MODE and MB_DATA_ADDR parameters

Description

Instead of a function code, the "MB_CLIENT" instruction uses the MB_MODE parameter. The MB_DATA_ADDR parameter is used to specify the Modbus start address of the data you want to access. The combination of the parameters MB_MODE and MB_DATA_ADDRdefines the function code used in the current Modbus message.

The following table shows the relationship between the MB_MODE parameter, the Modbus function and the address space.

MB_MODE	Modbus function	Data length	Function and data type	MB_DATA_ADDR
0	01	1 to 2000	Read output bits: 1 to 2000 bits per call	1 to 9999
0	02	1 to 2000	Read input bits: 1 to 2000 bits per call	10001 to 19999
0	03	1 to 125	Read holding register: 1 to 125 WORD per call	40001 to 49999
0	04	1 to 125	Read input words: 1 to 125 WORD per call	30001 to 39999
1	05	1	Write an output bit: One bit per call	1 to 9999
1	06	1	Write a holding register: 1 WORD per call	40001 to 49999
1	15	2 to 1968	Write multiple output bits: 2 to 1968 bits per call	1 to 9999

MB_MODE	Modbus function	Data length	Function and data type	MB_DATA_ADDR
1	16	2 to 123	Write several holding registers: 2 to 123 WORD per call	40001 to 49999
2	15	1 to 1968	Write one or more output bits: 1 to 1968 bits per call	1 to 9999
2	16	1 to 123	Write one or more holding registers: 1 to 123 WORD per call	40001 to 49999
11	11	0	Read status word and event counter of server communication: • The status word reflects the the processing status (0 - not processing, 0xFFFF - processing). • The event counter is incremented each time a message is sent successfully. The MB_DATA_ADDR and MB_DATA_LEN parameters of the "MB_CLIENT" instruction are not evaluated when this function executes.	-
80	08	1	Check the server status with the error code 0x0000 (return loop test - the server sends the request back): 1 WORD per call	-
81	08	1	Reset the event counter of the server with the error code 0x000A: 1 WORD per call	
3 to 10, 12 to 79, 82 to 255			Reserved	

See also

Description of MB_CLIENT (Page 1942)

MB_DATA_PTR parameter

Description

The MB_DATA_PTR parameter is a pointer to a data buffer for storing the data read from or written to the Modbus server. As the data buffer, you can use a global data block or a memory area (M).

For a buffer in the memory area (M), use a pointer in the ANY format as follows: "P#bit address" "data type" "length" (example: P#M1000.0 WORD 500).

The MB_DATA_PTR parameter uses a communications buffer:

- For the communication functions of the "MB_CLIENT" instruction:
  - Reading and writing of 1 bit of data of the Modbus server addresses 00001 to 09999 and 10001 to 19999.
  - Reading of 16-bit WORD data of the Modbus server addresses 30001 to 39999 and 40001 to 49999.
  - Writing 16-bit WORD data of the Modbus server addresses 40001 to 49999.
- During data transmission (length: bit or WORD) from or to the global DB or the memory area (M) that you assigned with the MB_DATA_PTR parameter.

If you use a data block for the buffer in the MB_DATA_PTR parameter, you will need to assign data types to the DB elements.

- Use the 1-bit data type BOOL for a Modbus bit address
- Use a 16-bit data type such as WORD, UINT, INT or REAL for a Modbus WORD address.
- Use a 32-bit data type (double word) such as DWORD, DINT or REAL for two Modbus WORD addresses.
- With MB_DATA_PTR, you can also access complex DB elements such as:
  - Standard arrays
  - Structures with unique element names
  - Complex structures with unique naming of the elements and data type lengths of 16 or 32 bits.
- The data areas for the MB_DATA_PTR parameter can also be in different global data blocks (or in different memory areas). You can, for example, use a data block for the the read jobs and another one for the write jobs or a separate data block for each "MB_CLIENT" station.

See also

Description of MB_CLIENT (Page 1942)

Parameter STATUS

STATUS parameter (protocol error)

STATUS (W#16#)	Code of the response to the Modbus client (B#16#)	Description
8381	01	Function code is not supported.
8382	03	Error in data length.
8383	02	Error in the data address or access outside the address area of MB_DATA_PTR (Page 1947).
8384	03	Error in data value.
8385	03	Error codes of diagnostics not supported (function code 08).

STATUS parameter (parameter error)

In addition to the errors listed in the following table, errors are also possible with the "MB_CLIENT" instruction caused by the communications instructions ("TCON", "TDISCON", "TSEND" and "TRCV").

STATUS (W#16#)	Description
80C8	No response of the server in the defined period. Check the connection to the Modbus server. This error is only reported on completion of the configured repeated attempts.
8188	The MB_MODE parameter has an invalid value.
818A	Invalid data length (MB_DATA_LEN parameter).
818B	The MB_DATA_PTR parameter has an invalid pointer. You should also check the values of the MB_DATA_ADDR (Page 1945) and MB_DATA_LEN parameters.
818C	The MB_DATA_PTR (Page 1947) pointer references an an optimized data block. Either use a data block with standard access or a memory area
8200	A further Modbus request is currently being processed via the port.
8380	Received block of transferred Modbus data is not well-formed or too few bytes were received
8387	• The assigned connection ID is different from that used for previous requests. Only one connection ID can be used for each instance DB of the "MB_CLIENT" instruction. • The error code is also output when the ID of the Modbus TCP protocol received by the server is not "0".
8388	The Modbus server sent a different data length than was requested. This error occurs only when using the Modbus functions 15 or 16.

See also

Description of MB_CLIENT (Page 1942)

MB_SERVER: Communicating via PROFINET as a Modbus TCP server

Description of MB_SERVER

Description

The "MB_SERVER" instruction communicates as a Modbus TCP server via the PROFINET connection of the S7-1200 CPU. To use the instruction, you do not require any additional hardware module. The "MB_SERVER" instruction processes connection requests of a Modbus TCP client, receives requests from Modbus functions and sends responses.

Parameters

The following table shows the parameters of the "MB_SERVER" instruction:

Parameters	Declaration	Data type	Description
DISCONNECT	Input	BOOL	The "MB_SERVER" instruction is used to enter into a passive connection with a partner module. The server reacts to a TCP connection request from every requesting IP address. • 0: Passive communication connection can be initialized • 1: Initialization of the connection termination. You can use this parameter to control when a connection request accepted. If the input to this parameter is set, no other operations are executed.
CONNECT_ID	Input	WORD	The parameter uniquely identifies a connection within the CPU. Each individual instance of the instructions "MB_CLIENT (Page 1942)" and "MB_SERVER" must have a unique CONNECT_ID parameter.
IP_PORT	Input	WORD	Start value = 502. The number of IP Ports defines which IP port is monitored for connection requests of the Modbus client. These TCP port numbers must not be used for the passive connection of the "MB_SERVER" instruction: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.
MB_HOLD_REG (Page 1952)	InOut	VARIANT	Pointer to the Modbus holding register of the "MB_SERVER" instruction: As the holding register, use either a global data block with standard access or a memory area (M). The holding register contains the values that may be accessed by a Modbus client using the Modbus functions 3 (read), 6 (write) and 16 (read).
NDR	Output	BOOL	"New Data Ready": • 0: No new data • 1: New data written by the Modbus client written
DR	Output	BOOL	"Data Read": • 0: No data read • 1: Data read by the Modbus client

Parameters	Declaration	Data type	Description
ERROR	Output	BOOL	If an error occurs during the call of the "MB_SERVER" instruction, the output of the ERROR parameter is set to TRUE. Detailed information about the cause of the problem is indicated by the STATUS parameter.
STATUS (Page 1953)	Output	WORD	Error code of the instruction.

For additional information on valid data types, refer to "Overview of the valid data types (Page 741)".

Mapping of Modbus addresses to the process image

The "MB_MASTER" instruction allows incoming Modbus functions (1, 2, 4, 5 and 15) direct read and write access to the process image input and output of the S7-1200 CPU (use of the data types BOOL and WORD).

For the data transfer of the function codes 3, 6 and 16, the holding register (MB_HOLD_REG parameter) must be defined longer than one byte. The following table shows the mapping of the Modbus addresses to the process image of the CPU.

Modbus function						S7-1200	
Code	Function	Data area	Address space			Data area	CPU address
01	Read: Bits	Output	1	to	8192	Process image output	Q0.0 to Q1023.7
02	Read: Bits	Input	10001	to	18192	Process image input	I0.0 to I1023.7
04	Read: WORD	Input	30001	to	30512	Process image input	IW0 to IW1022
05	Write: Bits	Output	1	to	8192	Process image output	Q0.0 to Q1023.7
15	Write: Bits	Output	1	to	8192	Process image output	Q0.0 to Q1023.7

Incoming Modbus messages with the function codes 3, 6 and 16 write to or read from the Modbus holding registers (you specify the holding registers with the MB_HOLD_REG parameter).

Multiple server connections

You can create multiple Server connections. This allows a single CPU to establish connections to more than one Modbus TCP client at the same time.

A Modbus TCP server can support several TCP connections and the maximum number of connections depends on the CPU being used.

The total number of connections of one CPU, including those of the Modbus TCP clients and server must not exceed the maximum number of supported connections.

Modbus TCP connections can also be shared by client and/or server connections.

In the case of Server connections, remember the following rules:

- Each "MB_SERVER" connection must use a unique instance DB.
- Each "MB_SERVER" connection must be created with a unique IP port number. Only one connection is supported for each port.
- Each "MB_SERVER" connection must use a unique connection ID.

  The relevant individual connection ID must be used for each individual instance DB of the instruction. The connection ID and instance DB belong together in pairs and must be unique for each connection.
- For each connection, the "MB_SERVER" instruction must be called individually.

Modbus diagnostics functions

The table below contains a description of the Modbus diagnostics functions.

Code	Subfunction	Description
08	0x0000	Echo test: The "MB_SERVER" instruction receives a data word and returns this unchanged to the Modbus master.
08	0x000A	Reset event counter: The "MB_SERVER" instruction resets the event counter for communication that is used for Modbus function 11.
11	-	Fetch event counter of the communication: The "MB_SERVER" instruction uses an internal event counter for communication to record the number of successfully executed read and write requests sent to the Modbus server. The event counter is not incremented by the functions 8, 11 or broadcast requests. The same applies to requests that result in a communications error (for example parity errors or CRC errors). The broadcast function is not available for Modbus TCP because only one client/server connection can exist at any one time.

Static variables of the instruction

The following table describes the static variables of the instance data block of the "MB_SERVER" instruction used in the program. You can write the HR_Start_Offset variable. You can read the other variables to monitor the Modbus status.

Variable	Data type	Start value	Description
HR_Start_Offset	WORD	0	Assign the start address of the Modbus holding register.
Request_Count	WORD	0	Total number of requests received by the server
Server_Message_ Count	WORD	0	Total number of received messages for the relevant Server.
Xmt_Rcv_Count	WORD	0	Counter for detecting the number of transfers during which an error occurred. The counter is also incremented if an invalid Modbus message is received.
Exception_Count	WORD	0	Counter for detecting the number of errors specifically for Modbus cause an exception.
Success_Count	WORD	0	Counter for detecting the number of requests that contain no error in the transferred protocol.

Example: Addressing via static variable HR_Start_Offset

The addresses of the Modbus holding register start at 40001. These addresses correspond to the address space of the CPU memory area for the holding register. You can also define the HR_Start_Offset variable so that the Modbus holding register has a start address other than 40001.

Example: The holding register begins at MW100, and has a length of 100 WORD. An offset value in the HR_Start_Offset parameter means that the start address of the holding register is moved from 40001 to 40021. Whenever the holding register is addressed below the address 40021 and above the address 40119, this causes an error.

HR_Start_Offset	Address	Minimum	Maximum
0	Modbus address (WORD)	40001	40099
	S7-1200 address	MW100	MW298
20	Modbus address (WORD)	40021	40119
	S7-1200 address	MW100	MW298

See also

MB_SERVER example: Multiple TCP connections (Page 1957)

MB_HOLD_REG parameter

Description

The MB_HOLD_REG parameter is a pointer to a data buffer for storing the data read from or written to the Modbus server. As the data buffer, you can use a global data block or a memory area (M).

As pointer to a buffer in the memory area (M), use the ANY format as follows: "P#bit address" "data type" "length" (example: P#M1000.0 WORD 500).

The following table shows examples of mapping Modbus addresses to the holding register for the Modbus functions 3 (read WORD), 6 (write WORD) and 16 (write several WORD). The upper limit for the number of addresses in the data block is decided by the maximum work memory of the CPU. If you use a memory area, the upper limit for the addresses is decided by the size of the memory area of the CPU.

Modbus addresses	MB_HOLD_REG parameter - examples		
	P#M100.0 WORD 5	P#DB10.DBx0.0 WORD 5	"Recipe".ingredient
40001	MW100	DB10.DBW0	"Recipe".ingredient[1]
40002	MW102	DB10.DBW2	"Recipe".ingredient[2]
40003	MW104	DB10.DBW4	"Recipe".ingredient[3]
40004	MW106	DB10.DBW6	"Recipe".ingredient[4]
40005	MW108	DB10.DBW8	"Recipe".ingredient[5]

See also

Description of MB_SERVER (Page 1949)

Parameter STATUS

Description

In addition to the errors listed in the following table, errors are also possible with the "MB_CLIENT" instruction caused by the communications instructions ("TCON", "TDISCON", "TSEND" and "TRCV").

STATUS (W#16#)	Code of the response to the Modbus server (B#16#)	Description
8187	No response	The MB_HOLD_REG parameter has an invalid pointer. Data area is too small.
818C	No response	• The MB_HOLD_REG parameter references an an optimized data block. Either use a data block with standard access or a memory area • Error due to timeout of execution (more than 55 seconds).
8381	01	Function code is not supported.
8382	03	Error in data length
8383	02	Error in data address or access outside the address area of the holding register (MB_HOLD_REG (Page 1952) parameter).
8384	03	Error in data value
8385	03	Value of the diagnostic code is not supported (only with function code 08).

See also

Description of MB_SERVER (Page 1949)

Examples

MB_CLIENT example 1: Send several requests via a TCP connection

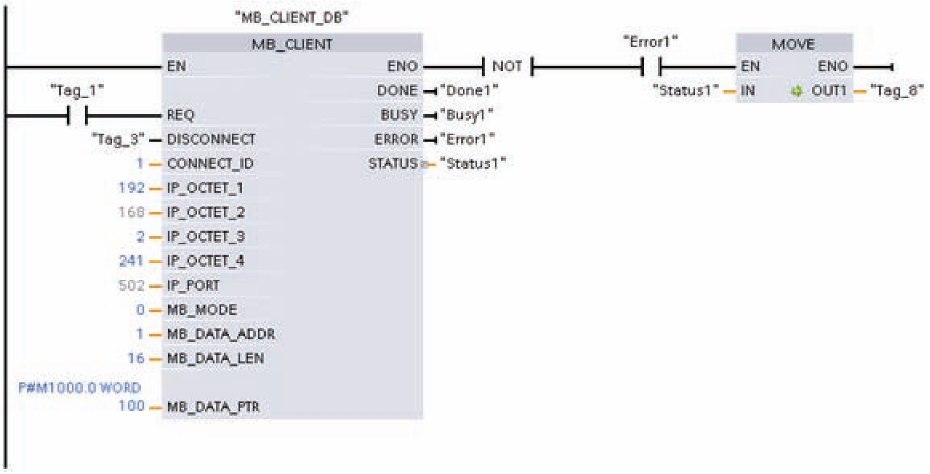
Description

Several requests of the Modbus Client can be sent via a TCP connection. To do this, use the same instance DB, the same connection ID and the same port number.

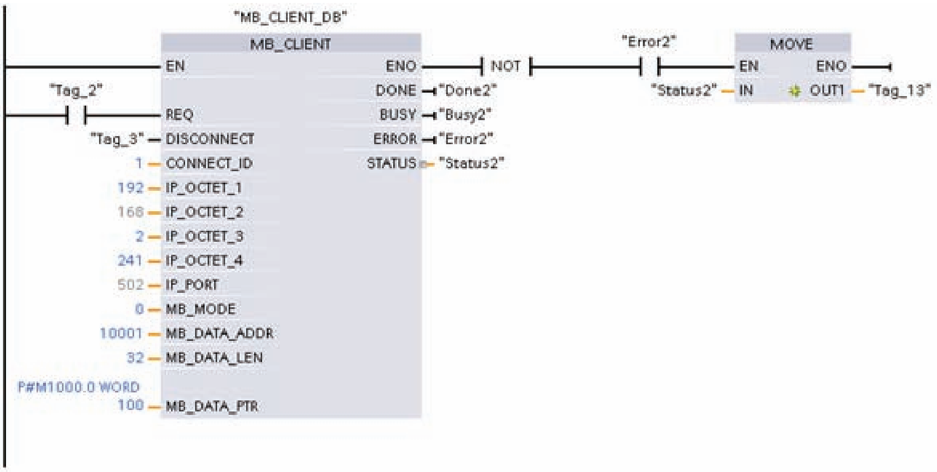
Only one client can be active at any one time. After processing of a client has been completed, the next client is processed. The order of execution must be defined in the program.

In the following sample program, the value of the STATUS output parameter is also copied.

Network 1: Modbus function 1 - 16 read output bits



Network 2: Modbus function 2 - 32 read input bits



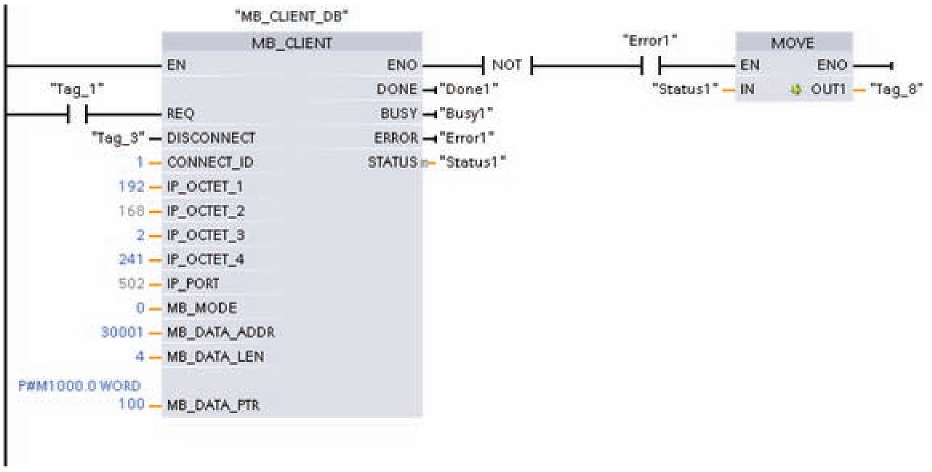
MB_CLIENT example 2: Send multiple requests via several TCP connections

Description

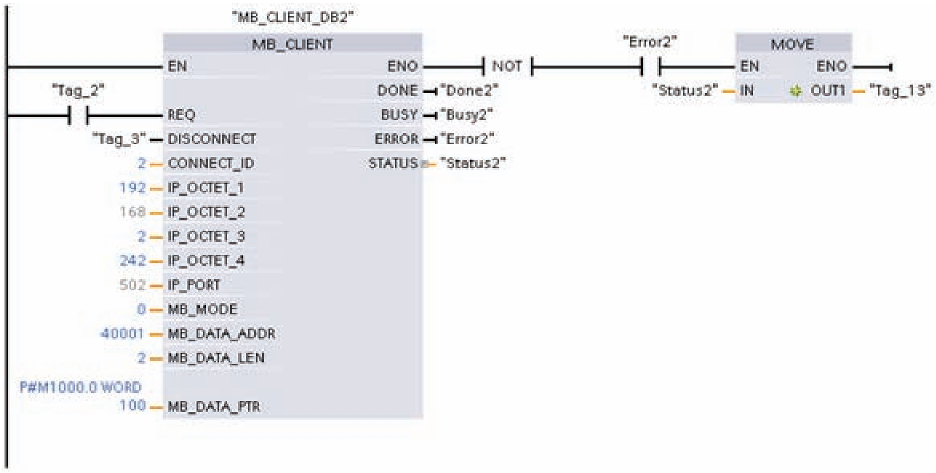
Requests from the Modbus client can be sent via different TCP connections. If you require this, use a different instance DB and a different connection ID.

Use a different port number, if the connections are to the same Modbus server. If the connections are to different Modbus servers, you can freely assign the port number.

Network 1: Modbus function 4 - read input (WORD)



Network 2: Modbus function 3 - read holding register (WORD)

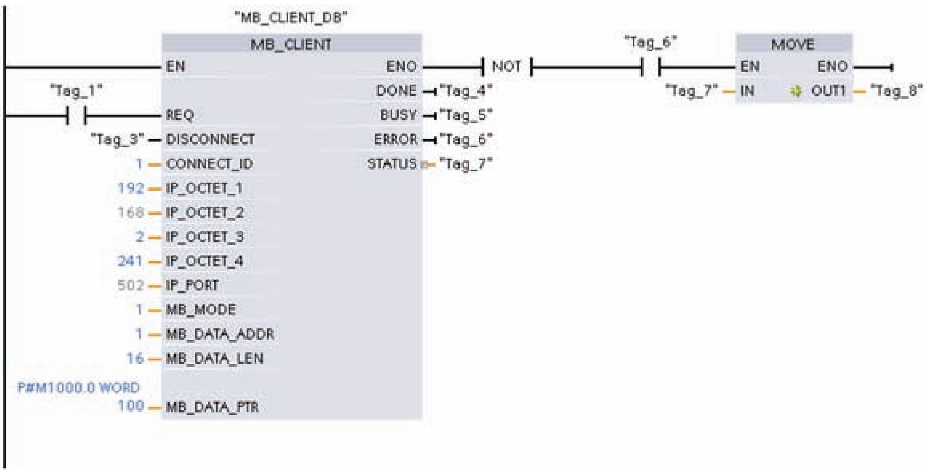


MB_CLIENT example 3: Respond to request to write from the Modbus server

Description

The following sample program reacts to requests of the Modbus server to write to the process image of the S7-1200.

Network 1: Modbus function 15 - write to process image of the S7-1200 (16 bits)

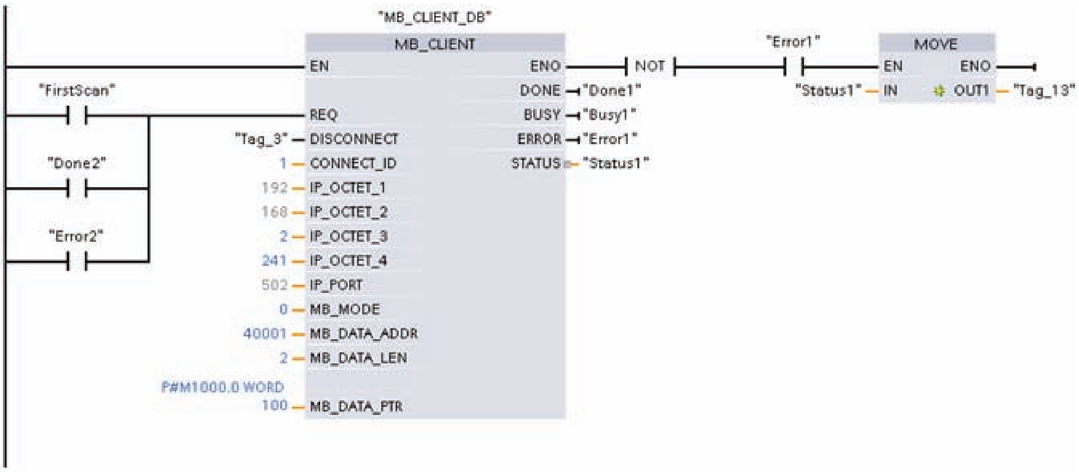


MB_CLIENT example 4: Coordinate several requests

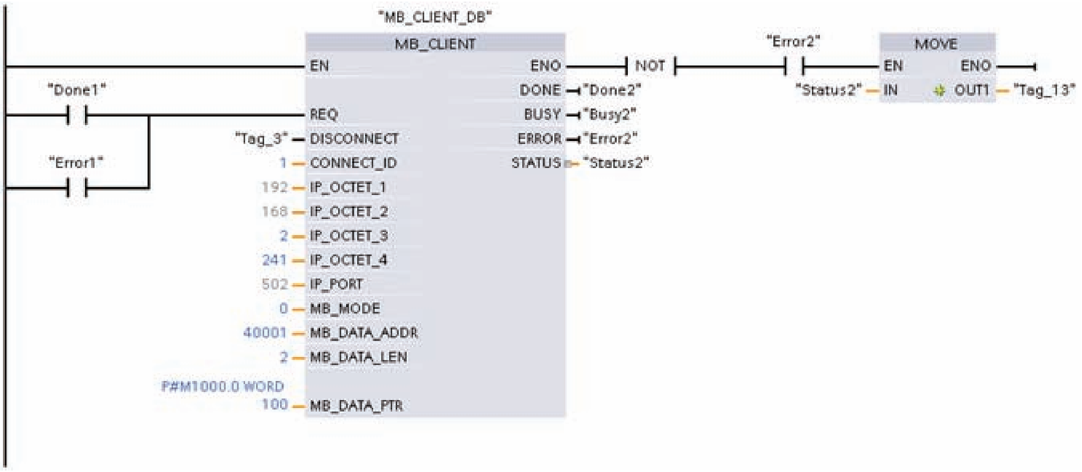
Description

Make sure that the individual Modbus requests are executed. You control coordination of requests with the program. The following example demonstrates how the output parameters of the first and second client request can be used to coordinate execution of the instructions.

Network 1: Modbus function 3 - read holding register (WORD)



Network 2: Modbus function 3 - read holding register (WORD)



MB_SERVER example: Multiple TCP connections

Description

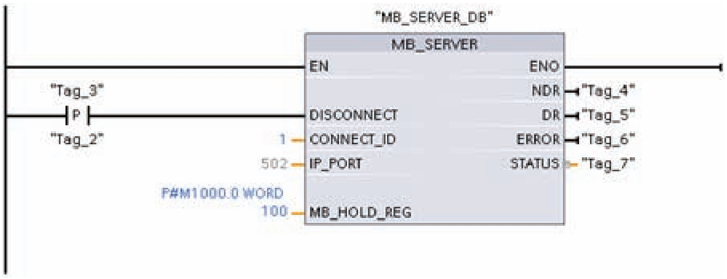
You can use several Modbus TCP server connections. To do this, the "MB_SERVER" instruction must be called independently for each connection.

Every connection requires the following:

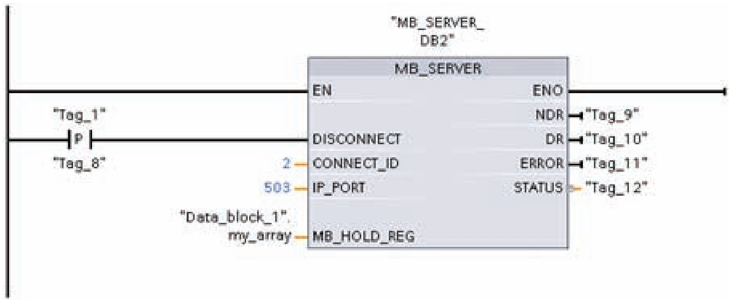
- An independent instance data block of the instruction
- A unique connection ID
- A separate IP port (on the S7-1200, only one connection is permitted per IP port)

To optimize the performance "MB_SERVER" should be executed once per program cycle for each connection.

Network 1: Connection #1 with associated IP port connection ID and instance DB



Network 2: Connection #1 with associated IP port connection ID and instance DB



9.8.5.5 TeleService

TM_MAIL: Transfer email

Description of TM_MAIL

Description

The "TM_MAIL" instruction works asynchronously, in other words, its execution extends over multiple calls. You must specify an instance when you call the instruction "TM_MAIL". The attribute "retentive" may not be set in the instance. This attribute ensures that the instance is initialized on the change of the CPU from STOP to RUN and that a new e-mail send job can be triggered afterwards.

You start the sending of an e-mail with an edge change from "0" to "1" for the REQ parameter. The job status is indicated by the output parameters BUSY, "DONE", "ERROR", "STATUS", and "SFC_STATUS". "SFC_STATUS" corresponds in this case to the "STATUS" output parameter of the called communication blocks.

The output parameters DONE, ERROR, STATUS, and SFC_STATUS are each displayed for only one cycle if the status of the BUSY output parameter changes from "1" to "0".

The following table shows the relationship between BUSY, DONE, and ERROR. Using this table, you can determine the current status of the instruction "TM_MAILand when the sending of the e-mail is complete.

DONE	BUSY	ERROR	Description
0	1	0	The job is being processed.
1	0	0	Job successfully completed.
0	0	1	The job was stopped with an error. The cause of the error can be found in the STATUS and SFC_STATUS parameters.
0	0	0	The "TM_MAIL" instruction was not assigned a (new) job.

If the CPU changes to STOP mode while "TM_MAIL" is active, the communication connection to the mail server aborts. The communication connection to the mail server will also be lost if communication problems occur on the Industrial Ethernet bus. In this case the transfer of the e-mail is interrupted and it cannot reach its recipient.

CAUTION

Changing user programs

You can change the parts of your user program that directly affect calls of "TM_MAIL" only:

- when the CPU is in "STOP" mode or
- when no mail is being sent (REQ = 0 and BUSY = 0).

This relates, in particular, to deleting and replacing program blocks that contain "TM_MAIL" calls or calls for the instance of "TM_MAIL".

Ignoring this restriction can tie up connection resources. The automation system can change to an undefined status for the TPC/IP communication functions via Industrial Ethernet.

A warm or cold restart of the CPU is required after the changes are transferred.

Data consistency

The ADDR_MAIL_SERVER input parameter of the instruction is taken from the "TM_MAIL" instruction again each time the sending of an e-mail is triggered. If a change is made during the operation, the "new" value only becomes effective after a renewed triggering of an e-mail.

In contrast, the WATCH_DOG_TIME, TO_S, CC, FROM, SUBJECT, TEXT, ATTACHMENT, and, if applicable, USERNAME and PASSWORD parameters are taken from the "TM_MAIL" instruction while it is running, which means that they may only be changed after the job is complete (BUSY = 0)

Setting the parameters of the TS Adapter IE

On the TS Adapter IE, you enter the parameters for the outgoing calls so that the TS Adapter IE can establish a connection to the dial-in server of your service provider.

If you set "When required" for connection establishment, the connection is first established when a mail needs to be sent.

Connection establishment can take a longer (approx. one minute) for an analog modem connection. When you specify the WATCH_DOG_TIME parameter, remember to allow for the time required to establish the connection.

Parameters

The following table shows the parameters of the "TM_MAIL" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter REQUEST: Activates the sending of an e-mail on a rising edge.
ID	Input	INT	D, L or constant	Reference to the connection to be established. See ID parameter of the TCON (Page 1868), TDISCON (Page 1871), TSEND (Page 1873), and TRCV (Page 1876) instructions. A number that is not used for any additional instances of this instruction in the user program must be entered here.
TO_S (Page 1962)	Input	STRING	D	Input parameter for receiver addresses: STRING with a maximum length of 240 characters (see example call).
CC (Page 1962)	Input	STRING	D	Input parameter for CC recipient addresses (optional): STRING with a maximum length of 240 characters (see example call). If an empty string is assigned here, the e-mail is not sent to a CC recipient.
SUBJECT	Input	STRING	D	Input parameter for subject of the e-mail. STRING with a maximum length of 240 characters.
TEXT	Input	STRING	D	Input parameter for text of the e-mail (optional): Reference to a data string with a maximum length of 240 characters. If an empty string is assigned at this parameter, the e-mail is sent without text.
ATTACHMENT	Input	VARIANT	I, Q, M, D, L	Input parameter for e-mail attachment: (optional): Reference to a byte/word/double word field with a maximum length of 65534 bytes. If no value is assigned, the e-mail is sent without an attachment.
DONE	Output	BOOL	I, Q, M, D, L	• DONE = 0: Job not yet started or still executing. • DONE = 1: Job was executed error-free.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The sending of the e-mail is not yet completed. • BUSY=0: The processing of "TM_MAIL" was stopped.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR=1: An error occurred during processing. STATUS and SFC_STATUS supply detailed information about the type of error.
STATUS (Page 1963)	Output	WORD	I, Q, M, D, L	Output/status parameter STATUS: Return value or error information of the "TM_MAIL" instruction.

Parameter	Declaration	Data type	Memory area	Description
ADDR_MAIL_ SERVER	Static*	DWORD	I, Q, M, D, L	IP address input parameter of the mail server: Specify as a data word in HEX format, for example: IP address = 192.168.0.200. ADDR_MAIL_SERVER = DW#16#C0A800C8, where: • 192 = 16#C0, • 168 =16#A8 • 0 = 16#00 and • 200 = 16#C8
WATCH_DOG_ TIME	Static*	TIME	I, Q, M, D, L	Input parameter for max. period of time: The "TM_MAIL" instruction should establish a connection within the period specified by WATCH_DOG_TIME. The block is exited with an error if this period is exceeded. The time before the block is exited and the error is output can exceed the WATCH_DOG_TIME because connection termination also takes a certain amount of time. To begin with, you should set a period of 2 minutes. If you have an ISDN telephone connection, a significantly shorter period can be selected.
USERNAME	Static*	STRING	D	Input parameter for user name: STRING with a maximum length of 180 characters. A user name is an absolute requirement for authentication.
PASSWORD	Static*	STRING	D	Input parameter for password: STRING with a maximum length of 180 characters. A password is an absolute requirement for authentication.
FROM (Page 1962)	Static*	STRING	D	Input parameter for sender address: STRING with a maximum length of 240 characters (see example call).
SFC_STATUS (Page 1963)	Static*	WORD	I, Q, M, D, L	Output/status parameter "SFC_STATUS": Error information of the called communication blocks.
*The values of the parameter are not modified at every call of the instruction "TM_MAIL". The values lie in the statistical parameters of the instance and are only described once at the first call of the instruction.				

You will find more detailed information on valid data types in "Overview of the valid data types (Page 741)".

Note

Optional parameters

The optional parameters CC, TEXT, and ATTACHMENT are only sent with the e-mail if the corresponding parameters contain a string of length > 0.

SMTP authentication

Authentication refers here to a procedure for ensuring an identity, for example, by a password query.

The "TM_MAIL" instruction supports the SMTP authentication procedure AUTH-LOGIN that is requested by most mail servers. For information about the authentication procedure of your mail server, refer to your mail server's manual or the Web site of your Internet service provider.

To use the AUTH-LOGIN authentication procedure, the "TM_MAIL" instruction requires the user name with which it can log onto the mail server. This user name corresponds to the user name with which you set up a mail account on your mail server. It is made available to the "TM_MAIL" instruction in the USERNAME parameter.

To log on, the "TM_MAIL" instruction also requires the associated password. This password corresponds to the password you specified when you set up your mail account. It is made available to the "TM_MAIL" instruction in the PASSWORD parameter.

If no user name is specified in the DB, the AUTH-LOGIN authentication procedure is not used. The e-mail is then sent without authentication.

Parameters TO_S, CC, and FROM

Description

The TO_S, CC, and FROM parameters are strings with, for example, the following content:

- TO_S: <wenna@mydomain.com>, <ruby@mydomain.com>,
- CC: <admin@mydomain.com>, <judy@mydomain.com>,
- FROM: <admin@mydomain.com>

Note the following rules when entering the parameters:

- The "TO_S:", "CC:", and "FROM:" characters must be entered.
- A space and an opening pointed bracket "<" must be entered before each address.
- A closing pointed bracket ">" must be entered after each address.
- A comma must be entered after each address in TO and CC.
- Only one e-mail may be entered in FROM, and there must not be a comma at the end of this address

Because of runtime and memory space, the "TM_MAIL" instruction does not perform any syntax check of the parameters TO_S, CC and FROM.

STATUS and SFC_STATUS parameters

Description

The return values of the "TM_MAIL" instruction can be classified as follows:

- W#16#0000: "TM_MAIL" was completed successfully
- W#16#7xxx: Status of "TM_MAIL"
- W#16#8xxx: An error was reported during the internal call of a communication block or from the mail server.

The following table shows the return values of "TM_MAIL" except for error codes of the internally called communication blocks.

Return value STATUS (W#16#...):	Return value SFC_STATUS (W#16#...):	Explanation	Notes
0000	-	The processing of "TM_MAIL" was completed without errors.	A error-free completion of "TM_MAIL" does not mean that the sent e-mail will necessarily arrive (see below - note on point 1)
7001		"TM_MAIL" is active (BUSY = 1).	Initial call; job has started
7002	7002	"TM_MAIL" is active (BUSY = 1).	Intermediate call; job already active
8xxx	xxxx	The processing of "TM_MAIL" was completed with an error code of the internally called communication instructions.	For detailed information on the evaluation of the SFC_STATUS parameter, refer to the descriptions of the STATUS parameter of the communication instructions.
8010	xxxx	Error during connection establishment.	For further information on the evaluation of the SFC_STATUS parameter, refer to the descriptions of the STATUS parameter of the "TCON (Page 1868)" instruction.
8011	xxxx	Error sending the data.	For detailed information on the evaluation of SFC_STATUS, refer to the descriptions of the STATUS parameter of the "TSEND (Page 1873)" instruction.
8012	xxxx	Error receiving the data.	For more information on the evaluation of SFC_STATUS, refer to the descriptions of the STATUS parameter of the "TRCV (Page 1876)" instruction.
8013	xxxx	Error during connection establishment.	For more information on the evaluation of SFC_STATUS, refer to the descriptions of the STATUS parameter of the "TCON (Page 1868)" and "TDISCON (Page 1871)" instructions.

Return value STATUS (W#16#...):	Return value SFC_STATUS (W#16#...):	Explanation	Notes
8014	-	Establishment of a connection is not possible.	You have possibly entered an incorrect mail server IP address (ADDR_MAIL_SERVER) or a time span that is too short (WATCH_DOG_TIME) to establish the connection. It is also possible that the CPU has no connection to the network or that the CPU configuration is incorrect.
82xx, 84xx, or 85xx	-	The error message originates from the mail server and corresponds, except for the "8", to the error number of the SMTP protocol. The following columns list several error codes that can occur:	See point 2 in the note.
8450	-	Action not executed: Mailbox not available/not reachable.	Try again later.
8451	-	Action aborted: Local processing error	Try again later.
8500	-	Syntax error: Error not recognized. This also includes the error when a command string is too long. This can also be occur when the e-mail server does not support the LOGIN authentication procedure.	Check the parameters of "TM_MAIL". Try to send an e-mail without authentication. To do this, replace the USERNAME parameter with an empty string.
8501	-	Syntax error: Parameter or argument incorrect	You have possibly entered an incorrect address in TO_S or CC.
8502	-	Command unknown or not implemented.	Check your entries, in particular the FROM parameter. This is possibly incomplete and you have forgotten "@" or ".".
8535	-	SMTP authentication incomplete.	You have possibly entered an incorrect user name or incorrect password.
8550	-	Mail server cannot be reached, you have no access rights.	You have possibly entered an incorrect user name or password, or the mail server does not support your LOGIN. Another cause of error could be an incorrect entry of the domain name after the "@" in TO_S or CC.
8552	-	Action aborted: Assigned memory size has been exceeded	Try again later.
8554	-	Transmission failed.	Try again later.

Note

Status error

1. An incorrect entry of the recipient addresses does not generate a status error of the "TM_MAIL" instruction. In this case, there is no guarantee that the e-mail will be sent to other recipients, even if these were entered correctly.
2. You will find more detailed information on the SMTP error code and other error codes in SMTP protocol on the Internet or in the error documentation of the mail server. You can also view the most recent message from the mail server in your instance DB in the BUFFER1 parameter. If you look under "Data", you will find the data most recently sent by the "TM_MAIL" instruction.

10 Visualizing processes (Basic)

10.1 Creating screens

10.1.1 Basics

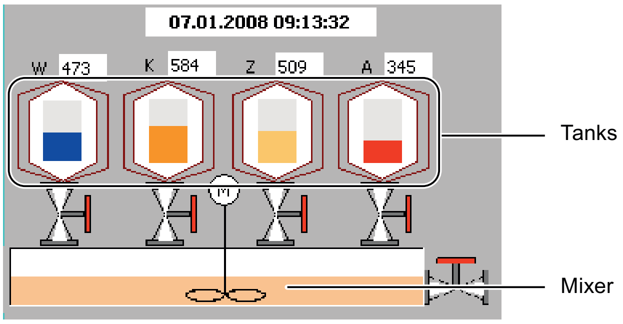
10.1.1.1 Screen basics

Introduction

In WinCC you create screens that an operator can use to control and monitor machines and plants. When you create your screens, the object templates included support you in visualizing processes, creating images of your plant, and defining process values.

Application example

The figure shows a screen that was created in WinCC. With the help of this screen, the operator can operate and monitor the mixing station of a fruit juice manufacturing system. Fruit juice base is supplied from various tanks to a mixing unit. The screen indicates the fill level of the tanks.



Screen design

Insert an object you need to represent a process into your screen. Configure the object to correspond to the requirements of your process.

A screen may consist of static and dynamic elements.

- Static elements such as text or graphic objects do not change their status in runtime. The tank labels (W, K, Z, A) shown in this example of a mixing unit are static elements.
- Dynamic elements change their status based on the process. Visual current process values as follows:
  - From the memory of the PLC
  - From the memory of the HMI device in the form of alphanumeric displays, trends and bars.

  Input fields on the HMI device are also considered dynamic objects. The fill level values of the tanks in our example of a mixing plant are dynamic objects.

Process values and operator inputs are exchanged between the controller and the HMI device via tags.

Screen properties

The screen layout is determined by the features of the HMI device you are configuring. It corresponds with the layout of the user interface of this device. The screen properties such as the screen resolution, fonts and colors are also determined by the characteristics of the selected HMI device. If the set HMI device has function keys, the screen shows these function keys.

A function key is a key on the HMI device. You can assign one or several functions in WinCC. These functions are triggered when the operator presses the relevant key on the HMI device.

A function key can be assigned global or local functions.

- Global function keys always trigger the same action, regardless of the currently displayed screen.
- Function keys with local assignment trigger different actions, based on the currently displayed screen on the operator station. This assignment applies only to the screen in which you have defined the function key.

Opening screens

In order for the operator to be able to call a screen in runtime, you must integrate each configured screen in the operating process. You have various options of configuring these functions:

- You use the "Screen" editor to configure buttons and function keys for opening other screens.
- You use the "Global Screen" editor to configure globally assigned function keys.

10.1.1.2 Availability of screens for specific HMI devices

Introduction

The functions of an HMI device determine the display of the device in WinCC and the scope of functions of the editors.

The following screen properties are determined by the functions of the selected HMI device:

- Device layout
- Screen resolution
- Number of colors
- Fonts
- Objects available

Device layout

The device layout of a screen forms the image of the HMI device in your configuration. The device layout of the screen shows all the function keys available on the HMI device, for example.

Screen resolution

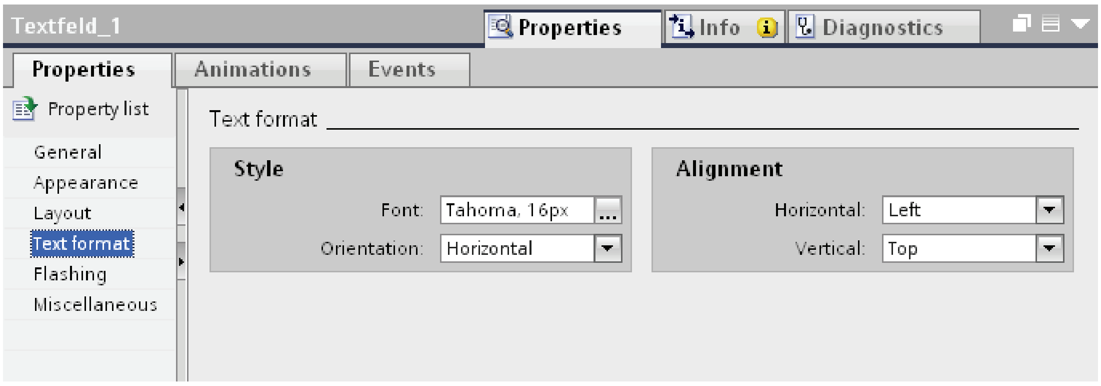
The screen resolution is determined by the different display dimensions of the various operator panels. You can only change the screen resolution if you configure the "WinCC Runtime Advanced" or "WinCC Runtime Professional" HMI device.

Number of colors

You can assign colors to the screen objects. The number of possible colors is determined by the color depth and specific colors supported on the selected HMI device.

Fonts

You can customize the appearance of the texts in all the screen objects that contain static or dynamic text. How to highlight individual texts in a screen. Select the font, font style and size, and set additional effects such as underscoring, for example.



Text format

Style

Font: Tahoma, 16px

Orientation: Horizontal

Alignment

Horizontal: Left

Vertical: Top

The settings for the text markups such as font style and effects always refer to the entire text of a screen object. That is, you can display the complete title in bold format, but not its individual characters or words, for example.

Objects available

Some of the screen objects can not be configured globally for all HMI devices. These screen objects are not displayed in the "Tools" task card. For a KTP 1000 touch panel unit you can not configure a slider, for example.

10.1.1.3 Basics

Task cards

Introduction

The following task cards are available in the "Screens" editor:

- Tools: Display and operating objects
- Animations: Templates for dynamic configuration
- Layout: Aid for customizing the display
- Libraries: Administration of the project library and of the global libraries

---

Note

WinCC Basic

The "Animations" task card is not available in WinCC Basic.

Tools

The "Tools" task card contains objects in different panes:

- Basic objects
- Elements
- Controls
- User controls (optional)
- Graphics

You paste objects from the palettes into your screens by drag&drop or a double click. The objects available for selection are determined by the features of the HMI device you are configuring. The following icons are used to change the display mode:

Icon	Meaning
▤	Displays the objects as a list.
▦	Displays the objects as a graphic.

Animations

The "Animations" task card contains the possible dynamizations of a screen object in the palettes. You paste the animations to a screen object by drag&drop or a double click from the "Movements", "Display" and "Tag Binding" palettes.

Layout

The "Layout" task card contains the following panes for displaying objects and elements:

- Zoom: Serves to select detail view
- Layers: Serves to manage screen object layers The layers are displayed in a tree view and contain information about the active layer and the visibility of all layers.
- Grid: You specify whether you want to align the objects to a grid or to other objects and set the grid size for a grid.
- Objects out of range: Objects that lie outside the visible area are displayed with name, position and type.

Libraries

The "Libraries" task card show the following libraries in separate panes:

- Project library: The project library is stored together with the project.
- Global library: The global library is stored in a separate file in the specified path on your configuration PC.

Move view

Introduction

You have the following options to display only a section of the entire screen in the work area:

- With the ▣ icon of the "Screens" editor.
- With the miniature view of the entire screen in the "Zoom" palette of the "Layout" task card.

Requirement

- A screen is open.
- The view shows only a screen section.

Procedure

To move the view:

1. Click the ▣ icon at the bottom right corner of the work area and press the left mouse button.

   A miniature view of the full screen is shown. An orange frame shows the currently selected area.
2. Hold down the mouse button and drag the frame to the desired area.

Note

The screen is scrolled when you drag a screen object from the visible to a currently hidden section.

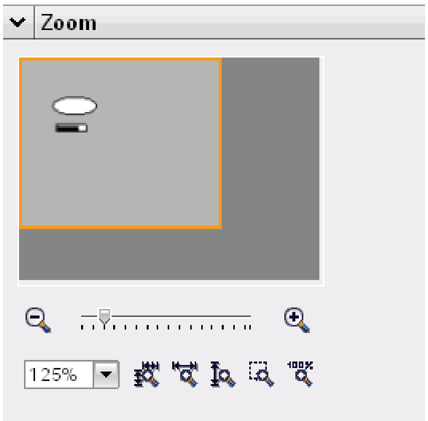
Zooming the view

Introduction

To view a small screen section in closer detail, use the zoom tool to magnify the screen in the working area. The maximum zoom amounts to 800%.

You can zoom with the toolbar in the work area or with the "Layout > Zoom" task card.

There are different ways to zoom the screen, e.g. with the zoom factor or by adapting the work area to the height of the screen.



Requirement

The screen is opened.

Procedure

Proceed as follows to zoom a view with the selection frame:

1. Click the toolbar button.
2. Use the mouse to draw a selection frame in the screen.

After you have released the mouse button, the section enclosed by the selection frame is zoomed to fit the complete work area.

Alternatively open the "Layout" task card and change the screen view.

Result

The selected screen section is magnified.

10.1.1.4 Working with screens

Steps

Steps

To create screens, you need to take the following initial steps:

- Plan the structure of the process visualization: Number of screens and their layout

  Example: Subprocesses are visualized in separate screens, and merged in a master screen.
- Define your screen navigation control strategies.
- Adapt the templates and the global screen.

  You define objects centrally and assign function keys for example.
- Create the screens. Use the following options of efficient screen creation:
  - Working with libraries
  - Working with layers
  - Working with faceplates

Creating a new screen

Introduction

Create screens to display processes in your system.

Requirement

- The project has been created.
- The Inspector window is open.

Procedure

1. Double-click "Screens > Add New Screen" in the project navigation.

   The screen is generated in the project and appears in your view. The screen properties are shown in the Inspector window.
2. Enter a meaningful name for the screen.
3. Configure the screen properties in the Inspector window:
   - Specify whether and on which template the screen is based.
   - Set the "Background Color" and the "Screen Number."
   - Specify a documenting text under "Tooltip".
   - Specify the layers to be displayed under "Layers" in the engineering system.
   - Select dynamic screen update under "Animations."
   - Select "Events" to define which functions you want to execute in Runtime when you call and exit the screen or at other events.

   Note

   Not all HMI devices support the "Visibility" animation.

Result

You created the screen in your project. You can now paste objects and control elements from the "Tools" task card and assign function keys in further work steps.

Managing Screens

Introduction

You can move screens within a project to other groups, or copy, rename, and delete them.

Moving screens in a group

1. Select the "Screens" folder in the project tree.
2. Select the "Add group" command from the shortcut menu.

   A folder called "Group_x" is inserted.
3. Select the screen in the project tree.
4. Drag-and-drop the screen to the required group.

   The screen is moved into this group.

Copy screen

1. Select the screen in the project tree.
2. Select the "Copy" command in the shortcut menu to copy the screen to the clipboard.
3. In the project tree, select the screen insert position.
4. Select "Paste" from the shortcut menu to insert the screen.

   A copy of the screen is inserted. A consecutive number is appended to the name of the original in the copy.

Alternatively, press <Ctrl> while you drag the screen to the required position.

Note

If you copy a screen with interconnected template for several devices and projects, the template will also be copied. Any existing matching template is not used. This holds particularly true when you copy the screens with drag-and-drop.

Rename screen

1. Select the screen in the project tree.
2. Select "Rename" from the shortcut menu.
3. Type in a new name.
4. Press <Enter>.

As an option, use the <F2> function key to rename the screen.

Delete screen

1. Select the screen in the project tree.
2. Select "Delete" from the shortcut menu.

   The screen and all its objects are deleted from the current project.

Defining the start screen of the project

Introduction

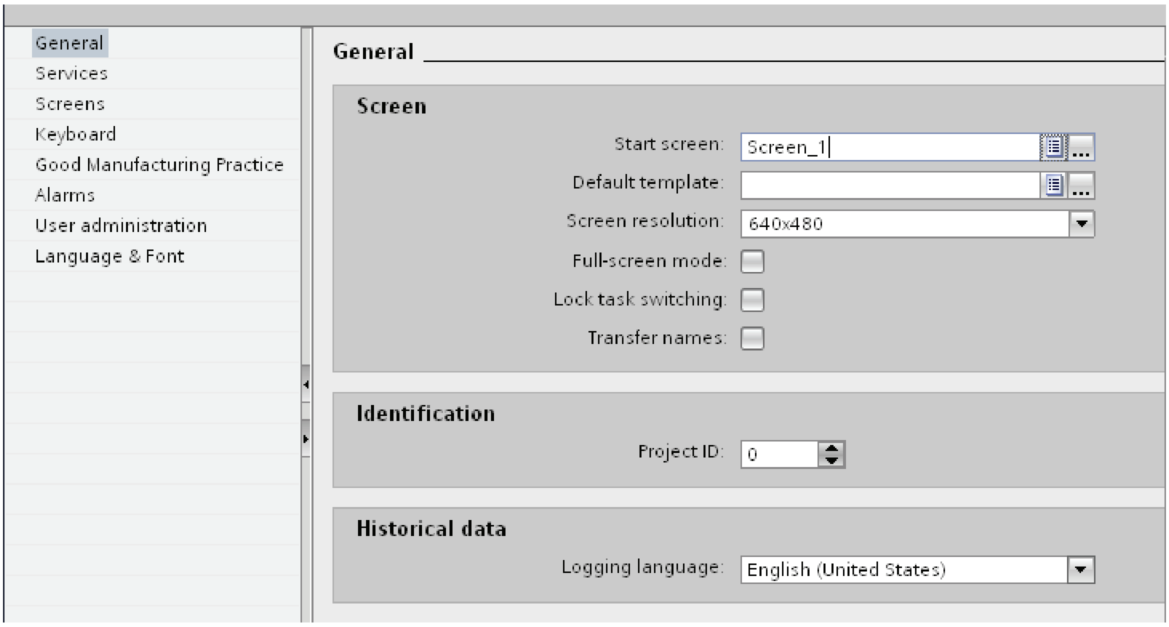
The start screen is the initial screen opened at the start of project in Runtime. You can define a different start screen for each one of the HMI devices. Beginning at this start screen, the operator calls the other screens.

Requirement

The project contains the screen you want to use as the start screen.

Procedure

1. Double-click "Runtime settings > General" in the project tree.



Start screen: Screen_1
Default template:
Screen resolution: 640x480
Full-screen mode: □
Lock task switching: □
Transfer names: □
Project ID: 0
Logging language: English (United States)

2. Select the desired screen as "Start screen."

Alternatively select a screen in the project tree and select "Use as start screen" in the shortcut menu.

Result

The start screen opens on the HMI at the start of Runtime.

10.1.1.5 Working with Templates

Basics on working with templates

Introduction

Configure the objects in a template which are to be displayed in all screens based on this template.

Note the following rules:

- A screen must not be based on a template.
- A screen is only based on one template.
- You can create several templates for one device.
- A template cannot be based on another template.

Objects for a template

You determine functions and objects in the template which are to apply for all screens based on this template:

- Assignment of function keys: You also assign the function keys in the template for HMI devices with function keys. This assignment overwrites a possible global assignment.
- Permanent window: Some devices support a permanent window for all screens in the top area of the screen. In contrast to the template, the permanent window occupies an area of the screen for itself alone.
- Operator controls: You can paste all screen objects which you also use for a screen into a template.

Application examples

- You want to assign the ActivateScreen" function to a function key in the template. The operator uses this key to switch to another screen in runtime. This configuration applies to all screens that are based on this template.
- A graphic with your company logo can be added to the template. The logo appears on all screens that are based on this template.

> **Note**
>
> If an object from the template has the same position as an object in the screen, the template object is covered.

See also

Creating a new template (Page 1980)

Managing templates (Page 1981)

Global screen (Page 1978)

Using a template in the screen (Page 1982)

Global screen

Introduction

You define global elements for all screens of an HMI device independently of the used template.

Function keys

For HMI devices with function keys you assign the function keys globally in the "Global Screen" editor. This global assignment applies for all screens of the HMI device.

Proceed as follows to assign function keys locally in screens or templates:

1. Click the function key in your screens or templates.
2. Deactivate "Properties > Properties > General > Use Global Assignment" in the inspection window.

Indicator and control objects for alarms

The "Alarm window" and "Alarm indicator" objects that are available as global objects are configured within the "Global screen" editor.

The "Alarm window" and "Alarm indicator" are always shown in the foreground.

For Comfort Panels you also have the possibility of configuring a "System diagnostic view" in the global screen.

Note

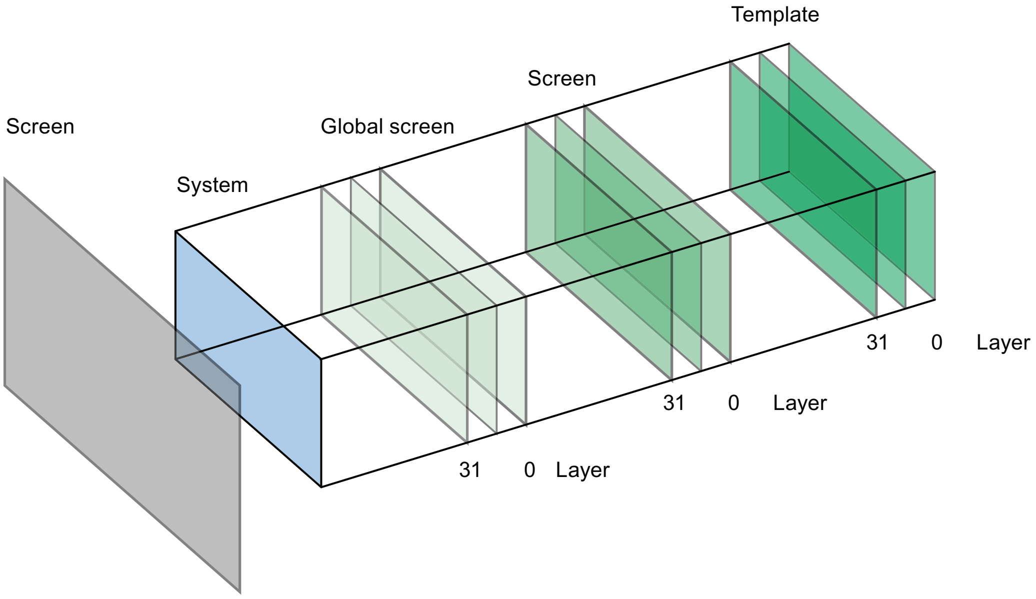
If you have configured a permanent window in a template, do not position the alarm window and alarm indicator in the area of the permanent window. Otherwise, the alarm window and the alarm indicator will not be displayed in Runtime.

The permanent window is not visible in the "Global screen" editor.

Order of configuration of screens

The following order applies for the configuration:

- The global screen comes before screens and templates
- Screens come before templates



The system layer is not configurable. This contains

- input dialog boxes
- alarms from the operating system
- the direct keys for touch panels

See also

Basics on working with templates (Page 1977)

Creating a new template

Introduction

In a template, you can centrally modify objects and function keys. Changes to an object or of a function key assignment in the template are applied to the object in all the screens which are based on this template.

Note

HMI device dependency

Function keys are not available on all HMI devices.

Requirement

- The project has been created.
- The Inspector window is open.

Procedure

1. Select "Screen management > Templates" in the project tree and then double-click "Add new template".

   The template is created in the project, and appears in your view.

   The properties of the template are displayed in the Inspector window.
2. Define the name of the template under "Properties > Properties > General" in the Inspector window.
3. Specify the layers in the engineering system that are displayed under "Properties >Properties >Layers" in the Inspector window.
4. Add the necessary objects from the "Tools" task card.
5. Configure the function keys.

Result

The template is created in your project.

See also

Basics on working with templates (Page 1977)

Managing templates

Introduction

You can move, copy, rename, and delete templates within a project in the Project window.

Moving a template into a group

1. Select the templates in the project navigation "Screen management > Templates".
2. Select "Add group" in the shortcut menu.

   A folder called "Group_x" is inserted.
3. Select the template in the project navigation.
4. Drag-and-drop the template to the required group.

   The template is moved to this group.

Copying templates

1. Select the template in the project navigation.
2. Select "Copy" in the shortcut menu.
3. Select the position in the project navigation where you want to paste the template.
4. Select "Paste" from the shortcut menu to insert the template.

   A unique name is assigned automatically to the copy.

Alternatively, you can hold down the <Ctrl> key, and drag the template into position.

Deleting a template

1. In the project navigation, select the template to be deleted.
2. Select "Delete" in the shortcut menu.

   The template, and all its objects are deleted from the current project.

Assigning a template to a screen

1. In the project navigation, select the screen to which you want to assign the template.
2. In the Inspector window, select "Properties > Properties > General".
3. Select the desired template under "Template."

   The selected template and all the objects contained in it are assigned to the screen.

See also

Basics on working with templates (Page 1977)

Using a template in the screen

Introduction

You use a template in the screen. All the objects configured in the template are also available in the screen.

Requirement

A template has been created.

A screen has been created.

Procedure

Proceed as follows to use a template in a screen:

1. Double click a screen in the project tree. The screen opens in the work area.
2. Open "Properties > Properties > General" in the inspector window.
3. Select a template that is to be applied to the screen under "Template".

Show template in screen

When you edit a screen, you can show an existing template in the screen.

Proceed as follows to show a template in the screen:

1. Activate "Extras > Settings > Visualization > Show template in screens" in the menu.

Result

The screen is based on the selected template. All objects which you have configured in the template are available in the screen. The template is displayed in the screen.

See also

Basics on working with templates (Page 1977)

10.1.2 Working with objects

10.1.2.1 Overview of objects

Introduction

Objects are graphics elements which you use to design the screens of your project.

The "Tools" task card contains all objects that can be used for the HMI device. Display the Task Card with menu command "View" by activating the "Task Card" option.

The toolbox contains various palettes, depending on the currently active editor. If the "Screens" editor is open, the toolbox contains the following palettes:

- "Basic objects"

  The basic objects include basic graphic objects such as "Line", "Circle", "Text field" or "Graphic view".
- "Elements"

  The elements include basic control elements, e.g. "I/O field", "Button" or "Gauge".
- "Controls"

  The controls provide advanced functions. They also represent process operations dynamically, for example Trend view and Recipe view.

- "Graphics"

  Graphics are broken down into subjects in the form of a directory tree structure. The various folders contain the following graphic illustrations:

  – Machine and plant areas

  – Measuring equipment

  – Control elements

  – Flags

  – Buildings

  You can create links to your own graphic folders. The external graphics are located in these folders and subfolders. They are displayed in the toolbox and incorporated into the project using links.

- "Libraries" task card

  In addition to the display and operating elements, the library objects are available. They are located within the palettes of the "Libraries" task card. A library contains preconfigured objects such as graphics of pipes, pumps, or preconfigured buttons. You can also integrate multiple instances of library objects into your project without having to reconfigure them.

  The WinCC software package includes libraries, e.g. "HMI Buttons & Switches".

  You can also store customized objects, and faceplates in user libraries.

  Faceplates are objects that you create from existing screen objects, and for which you define the configurable properties.

Note

HMI device dependency

Some of the toolbox objects are either available with restricted functionality, or not at all. This depends on the HMI device you are configuring. Unavailable properties of an object are displayed as deactivated, and cannot be selected.

Basic objects

Icon	Object	Instructions
╱	"Line"	-
⬬	"Ellipsis"	-
●	"Circle"	-
▬	"Rectangle"	-

Icon	Object	Instructions
A	"Text field"	One or more lines of text. The font and layout are adjustable.
	"Graphic view"	Displays graphics from external graphic programs, and inserts OLE objects. The following graphic formats can be used: "*.emf", "*.wmf", "*.dib", "*.bmp", "*.jpg", "*.jpeg", "*.gif" and "*.tif".

Elements

Icon	Object	Instructions
0.12	"I/O field"	Outputs the values of a tag, and/or writes values to a tag. You can define limits for the tag values shown in the I/O field. To hide the operator input in Runtime, activate "Hidden input."
	"Button"	Executes a list of functions, or a script as configured.
	"Symbolic I/O field"	Outputs the values of a tag, and/or writes values to a tag. A text from a text list is displayed in relation to the tag value.
	"Graphic I/O field"	Outputs the values of a tag, and/or writes values to a tag. A graphic from a graphics list is displayed in relation to the tag value.
5	"Date/time field"	Outputs the system date and time, or the time and date from a tag. This allows the operator to enter new values. The display format is adjustable.
	"Bar"	The bar represents a value from the PLC in the form of a scaled bar graph.
0 I	"Switch"	Toggles between two defined states. You can label a switch with text, or a graphic.

Controls

Icon	Object	Description
⚠	"Alarm view"	Shows currently pending alarms or alarm events from the alarm buffer or alarm log.
	"Trend view"	Represents multiple curves with values from the PLC, or from a log.
	"User view"	Allows an administrator to administer users on the HMI device. It also allows an operator without administrator rights to change his password.
	"Recipe view"	Displays data records, and allows them to be edited.

See also

Options for Editing Objects (Page 1986)

Inserting an object (Page 1987)

Deleting an Object (Page 1989)

Positioning an object (Page 1990)

Resizing an object (Page 1991)

Selecting multiple objects (Page 1992)

Aligning objects (Page 1995)

Moving an object forward or backward (Page 1997)

Show objects outside the screen area (Page 1998)

Rotating objects (Page 1998)

Flipping objects (Page 2000)

Inserting multiple objects of the same type (stamping tool) (Page 2001)

Repositioning and resizing multiple objects (Page 2003)

External graphics (Page 2004)

Managing external graphics (Page 2005)

Storing an external image in the graphics library. (Page 2007)

Basics on groups (Page 2009)

Overview of keyboard access (Page 2015)

Example: Inserting and configuring a rectangle (Page 2019)

10.1.2.2 Options for Editing Objects

Introduction

Objects are graphics elements which you use to design the screens of your project.

You have the following options for editing objects:

- Copying, pasting or deleting objects using the shortcut menu If you copy an object in a screen and the screen already includes an object of the same name, the name of the object is changed.
- Maintaining the standard size of the objects you are inserting or customizing their size on insertion.
- Changing the properties of an object, e.g. the size
- Positioning an object
- Moving an object in front of or behind other objects
- Rotating objects
- Mirroring objects

- Changing default properties of the objects
- Defining the tab sequence for objects
- Stamping: Inserting several objects of the same type
- Selecting several objects simultaneously
- Repositioning and resizing multiple objects
- You can assign external graphics to objects, e.g. in the Graphic View.

  You can view only the images you have previously stored in the graphic browser of your WinCC project.

  You can save graphics in the graphic browser:

  – Via drag & drop from the "Graphics" object group to the working area

  – As graphic files in the following formats: *.bmp, *.dib, *.ico, *.emf, *.wmf, *.gif, *.tif, *.jpeg or *.jpg

  – As an OLE object

    You either create a new OLE object or save an existing graphic file as an OLE object. To save an OLE object, an OLE-compatible graphics program must be installed on the configuration computer.

See also

Overview of objects (Page 1983)

10.1.2.3 Inserting an object

Introduction

In the "Screens" or "Reports" editor, insert the objects to the "Toolbox" task card. Use the mouse to drag the objects into the work area. You either keep the objects in their original size, or scale them up or down when you paste them.

In addition, you can copy or move objects via the clipboard from one editor to another, for example to transfer a screen object to a report. Alternatively, you can also use the mouse instead of the clipboard for copying and moving:

- Copying: <Ctrl + Drag&Drop>
- Moving: Drag&drop

Note

Basic Panels

The "Reports" editor is not available for Basic Panels.

Requirement

The "Tools" task card is open.

Inserting objects in their standard size

1. In the "Toolbox" task card, select the desired graphic object or the desired graphic in the WinCC graphics folder.

   When you move the cursor across the work area, it turns into a crosshair with an appended object icon.
2. Click the location in the work area where you want to insert the object or graphic.

   The object is inserted with its standard size at the desired position in the work area.

Alternatively, double-click the object in the "Toolbox" task card.

Copying an object

1. Select the desired object.
2. Select "Copy" in the shortcut menu.
3. Click the desired location and select "Paste" in the shortcut menu.

WinCC inserts a copy of the object at the desired location. You can only change the properties that are appropriate for the relevant context.

Example: In the "Screens" editor, you can set for I/O fields and the mode for input and output. In the "Reports" editor, the mode is set to "Output".

Original and copy are not interconnected and are configured independently from one another.

Inserting lines

1. Select the desired graphic object in the "Tools" task card.
2. Click on a location in the work area. A line in the standard size is inserted.

Inserting a polygon or polyline

1. Select the desired object "Polyline" or "Polygon" in the "Tools" task card.
2. Click on a location in the work area. This fixes the starting point of the object.
3. Click another location in the work area. A corner point is set.
4. For every additional corner point, click on the corresponding location in the work area.
5. Double-click on a location in the work area. The last corner point is set.

   This fixes all the points of the polygon or polyline.

Note

Basic Panels

The "Polyline" and "Polygon" objects are not available for Basic Panels.

Note

If you want to insert several objects of the same type, use the "Stamp" function. This avoids having to reselect the object in the "Tools" task card every time before inserting it. To do so, select the icon in the toolbar of the "Tools" task card.

See also

Overview of objects (Page 1983)

10.1.2.4 Deleting an Object

Introduction

You can delete objects individually or with a multiple selection.

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object that you want to delete.

   To delete multiple objects, keep the <Shift> key pressed and select the objects to be deleted one after the other. Alternatively, drag and maximize an area around the desired objects with the mouse.
2. Select "Delete" from the shortcut menu.

Result

The selected objects are deleted.

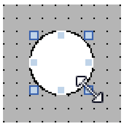
See also

Overview of objects (Page 1983)

10.1.2.5 Positioning an object

Introduction

When you select an object, it is enclosed by a rectangle with resizing handles. This rectangle is the rectangle which surrounds the object. The position of an object is defined by the coordinates of the top left corner of the rectangle surrounding the object.



Note

If the position is outside the work area the object is not displayed in Runtime.

Position and align

You have the possibility of having a grid displayed in the work area. You have three options for easier positioning of objects:

- "Snap to grid" When you resposition objects, they are automatically snapped and pasted to the grid. If you hold down the <Alt> key, the object is no longer snapped to the grid.
- "Snap to objects" When you reposition objects, they are displayed with help lines. You can use other objects for orientation during positioning.
- "None": You position the objects in any position.

You activate and deactivate the grid and options as follows:

- In the "Options > Settings > Visualization > Screens" menu
- In the "Layout > Grid" task card

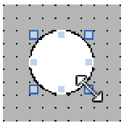
Requirement

You have opened the work area containing at least one object.

Procedure

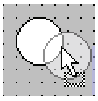
1. Select the object you want to move.

   The selected object is framed by a rectangle with resizing handles.

2. Left-click the object and keep the mouse button pressed.
3. Move the mouse pointer onto the new position.

   The contour of the object moves with the mouse and displays the new position for the object.

   The object initially remains at its original position.
4. Now release the mouse button.

   The object is moved into the position indicated by the contour of the selection rectangle.

Alternative procedure

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the X and Y values for the position under "Position & Size".

Result

The object appears at its new position.

See also

Overview of objects (Page 1983)

10.1.2.6 Resizing an object

Introduction

When you select an object, it is enclosed by a rectangle with handles. You have the following options of resizing an object:

- Drag the handles using the mouse.
- Modify the "Size" property in the Inspector window.

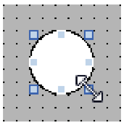
Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object you want to resize.

   The selection rectangle appears. The following figure shows a selected object:

2. Drag a resizing contact of the rectangle to a new position.

   The size of the object changes.

   – The size of the object is aligned to the grid pattern, provided the "Snap to grid" function is set.

   – Press <ALT> to disable this function while you drag the object.

   Note

   In order to scale the object proportionally, keep the <Shift> key pressed while changing the size with the mouse.

Alternative procedure

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the size of the object under "Position & Size".

Result

The object now appears with its new size.

See also

Overview of objects (Page 1983)

10.1.2.7 Selecting multiple objects

Introduction

Select all objects you want to align with each other or to change global properties. This procedure is called "multiple selection."

The Inspector window shows all the properties of the selected objects.

You now have several options of selecting multiple objects:

- Draw a selection frame around the objects.
- Hold down the <Shift> key, and click the required objects.

Selection frame of a multiple selection

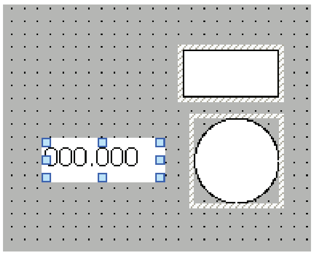
The selection frame surrounds all objects of a multiple selection. The selection frame is comparable with the rectangle that surrounds an individual object.

The selection frame is not visible. When you have made your multiple selection, the following frame is displayed:

- The reference object is indicated by the rectangle around it.
- The other selected objects are indicated by a dashed-line frame.

Specifying a reference object

The reference object is the object upon which the other objects are oriented. The reference object is framed by a rectangle with handles. The following figure shows a reference object with two other selected objects:



You have the following options to specify the reference object:

- Select the objects via multiple selection. The object selected first is then the reference object.
- Draw a selection frame around the objects. The reference object compiled automatically. If you wish to specify a different object within the selection as the reference object, click on the desired object. This action does not cancel your multiple selection.

Requirement

You have opened the work area containing at least two objects.

Selecting multiple objects with a selection frame

1. Position the mouse pointer in the work area close to one of the objects to be selected.
2. Hold down the mouse button, and draw a selection frame around the objects to be selected.

Or:

1. Hold down the <Shift> key.
2. Click the relevant objects, working in succession.

   All the selected objects are identified by frames.

   The object selected first is identified as reference object.

   Note

   To remove an object from the multiple selection, press <SHIFT>, hold it down and then click the relevant object once again.

Result

Multiple objects are selected. One of those is identified as the reference object. You can now perform the following steps:

- Changing the object properties of all the objects
- Resizing all the objects by the same ratio, by dragging the selection frame to increase or reduce the size
- Moving all the objects in one group
- Aligning the objects to the reference object

See also

Overview of objects (Page 1983)

10.1.2.8 Aligning objects

Procedure

1. Select the objects via multiple selection.
2. Specify an object as the reference object.
3. Select the desired command in the toolbar or the shortcut menu - see table below.

   The selected objects will be aligned.

Aligning objects flush

The selected objects will be aligned flush to the reference object.

Icon	Description
	Aligns the selected objects to the left edge of the reference object.
	Aligns the selected objects to the vertical center axis of the reference object.
	Aligns the selected objects to the right edge of the reference object.
	Aligns the selected objects to the upper edge of the reference object.
	Aligns the selected objects to the horizontal center axis of the reference object.
	Aligns the selected objects to the lower edge of the reference object.
	Centers the selected objects to the center points of the reference object.
	Centers the selected objects vertically in the screen.

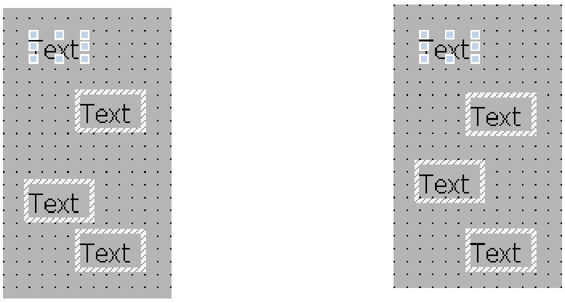
Distributing objects evenly

You need at least three selected objects. A reference object is not required.

1. Select the objects.
2. Click one of the buttons "Distribute horizontally equal" or "Distribute vertically equal."

   The selected objects are distributed at equal distances.

The following screen shows how you align the vertical spacing of the selected objects:



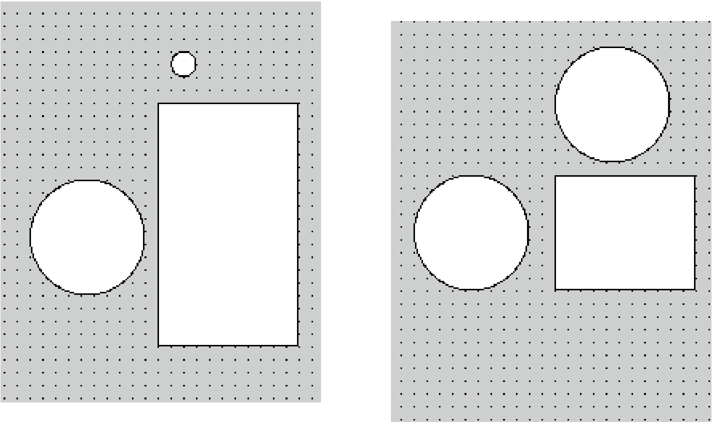
Icon	Description
	Aligns the horizontal distance between the objects. The position of the objects on the extreme left and right side remains unchanged. All other objects are distributed evenly between them.
	Aligns the vertical distance between the objects. The position of the objects at the extreme top and bottom (right and left) remains unchanged. All other objects are distributed evenly between them.

Harmonizing the object size

1. Select the objects.
2. Now, click one of the following buttons: or or

   The size of the selected objects is matched to each other.

The following screen shows how the selected objects are adapted to the height of the reference object:



Icon	Description
	Aligns the selected objects to the width of the reference object.
	Aligns the selected objects to the height of the reference object.
	Aligns the selected objects to the width and height of the reference object.

See also

Overview of objects (Page 1983)

10.1.2.9 Moving an object forward or backward

Introduction

You can use the "Order" functions in the shortcut menu of a selected object or in the toolbar to move a selected object in front of or behind other objects within an object layer.

Note

ActiveX controls are always positioned in front of an object layer (.NET property).

Requirement

You have opened a screen which contains a layer with multiple objects.

Procedure

1. Select the object you want to move forward or backward.
2. Select the "Sort" command and one of the following commands from the shortcut menu:

Icon	Description
	Moves the selected object before all the other objects of the same layer
	Moves the selected object to the lowest position in the same layer
	Moves the selected object up by one position
	Moves the selected object down by one position

Alternative procedure

1. Open the "Layers" palette of the "Layout" task card.
2. Navigate to the required object.
3. Hold down the mouse button, and drag the object in the tree topology to the required position in the layer.
4. Now release the mouse button.

Result

The object is moved up or down.

See also

Overview of objects (Page 1983)

10.1.2.10 Show objects outside the screen area

Introduction

If you assign objects to positions that are outside the configurable area, these objects will be hidden. The functions of the "Objects outside the visible area" palette in the "Layout" task card are used to move these objects back into the screen.

Requirement

- You have opened a screen which contains objects that are outside the configurable area.
- The "Layout" task card is open.

Procedure

1. Open the "Layout > Objects outside the area" task card.

   This displays a list of objects that are outside the configurable area.
2. Select the the object which you want to move back into the screen from the list.
3. Select "Move to screen" in the object shortcut menu.

Alternatively open the "Layout > Layer" task card. Objects outside the area are indicated by the icon. If you click this icon, the object is moved back into the screen.

Result

The objects are moved to the configurable area.

See also

Overview of objects (Page 1983)

10.1.2.11 Rotating objects

Introduction

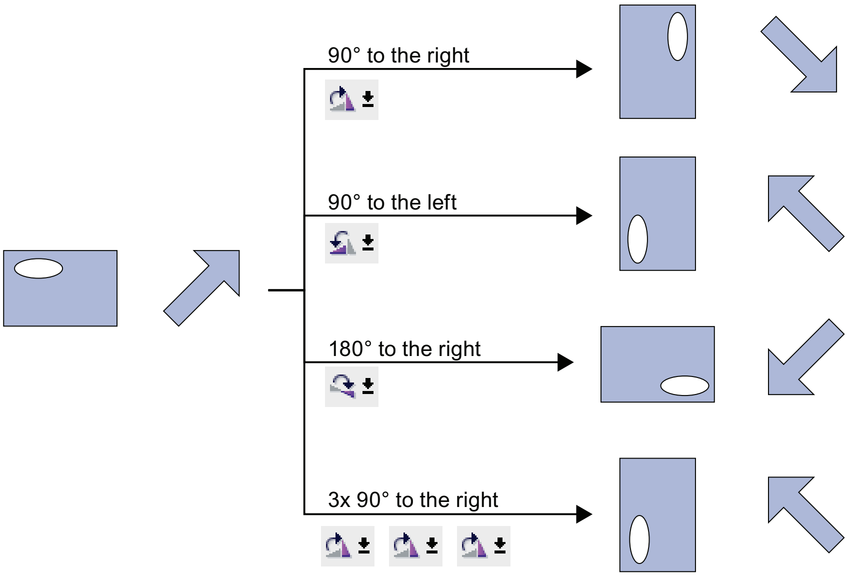
You can rotate a suitable object clockwise or counterclockwise around its center axis in steps of 90°.

Note

Not all the objects can be rotated. Some objects that can be rotated in screens cannot be rotated in reports.

You can also rotate multiple objects using the multiple selection function. Certain WinCC objects such as buttons cannot be rotated.

The alignment of elements in an object will change in a rotated object. The following figure shows how a rectangle and an ellipse behave under the different commands for rotating an object:



Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object that you want to rotate.
2. Click one of the following toolbar icons:

   ↻, to rotate the object clockwise around its center point. The angle of rotation is 90°.

   ↺, to rotate the object counterclockwise around its center point. The angle of rotation is 90°.

   ↻, to rotate the object clockwise by 180°.

Result

The object is shown at its new angle.

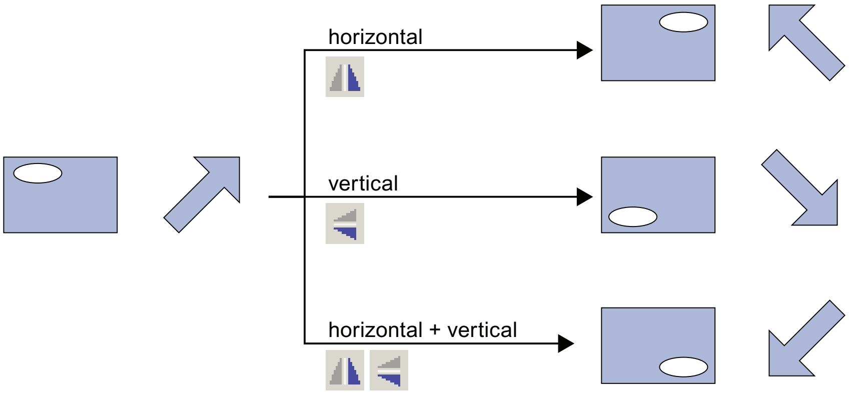
See also

Overview of objects (Page 1983)

10.1.2.12 Flipping objects

Introduction

You can flip an object along its vertical or horizontal center axis. The alignment of elements in an object will change when you flip an object. The following figure shows how a rectangle and an ellipse behave under the different commands for flipping an object.



Requirement

You have opened a screen which contains at least one object.

Procedure

1. Select the object that you want to flip.
2. Click the "Flip" command in the shortcut menu and select one of the options displayed:
   - ◭ ±, to flip the selected object along its vertical center axis.
   - ◀ ±, to flip the selected object along its horizontal center axis.

Result

The object is shown at its flipped position.

See also

Overview of objects (Page 1983)

10.1.2.13 Designing an object

Introduction

You design the border and background of an object.

Requirement

A line has been created in a screen.

Procedure

1. Select the line on your screen.
2. In the Inspector window, select "Properties > Properties > Appearance":
3. Select "Dash" as the style.
4. To display the dashed line in two colors, select the line width "1".
5. Select the setting "Arrow" in the "Line ends" area.

Result

The line is displayed in two colors as a dashed line. The end of the line is an arrow.

10.1.2.14 Inserting multiple objects of the same type (stamping tool)

Introduction

WinCC offers the possibility to "stamp" several objects of the same type directly one after the other, i.e. paste without having to reselect the object every time. In addition you have the possibility of multiplying an object that has already been inserted.

Requirement

The "Tools" task card is open.

Inserting several objects of one type

1. Select the object that you want to insert in the "Tools" task card.
2. Click the icon in the toolbar of the "Tools" task card.

   The "Stamp" function is activated.
3. To insert the object with its standard size, click the relevant insertion position in the work area.

   To insert the object with another size, position the mouse pointer at the desired location in the work area. Press the left mouse button and drag the object to the required size.

   The object is inserted in the work area as soon as you release the mouse button.
4. Repeat step 3 to insert further objects of the same type.
5. Click the icon again.

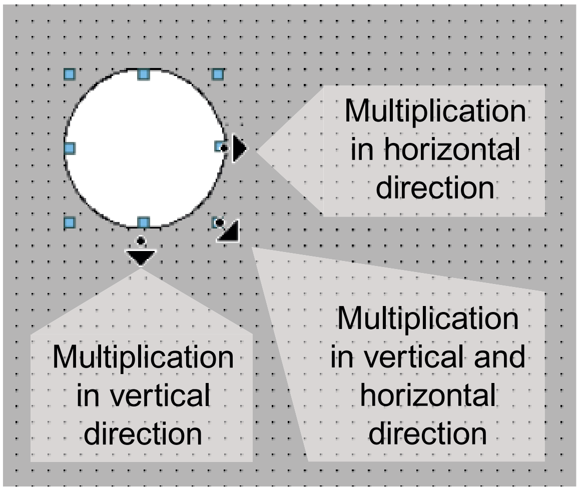
   The "Stamp" function is deactivated.

Note

You can copy existing objects using the drag-and-drop +<CTRL> function. The existing object is not moved in this case. You paste a copy of this object into the new position instead.

Inserting and multiplying an object

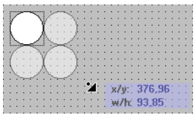
1. Insert the desired object from the "Tools" task card.
2. Press the <Ctrl> key and position the cursor on one of the handles displayed in the figure shown below.


Multiplication in horizontal direction
Multiplication in vertical direction
Multiplication in vertical and horizontal direction

3. Drag the handles to the right and/or down while keeping the left mouse button pressed.
4. The object is multiplied depending on available space if you keep moving the cursor.

Result

You have pasted and stamped an object in a screen.



See also

Overview of objects (Page 1983)

10.1.2.15 Repositioning and resizing multiple objects

Possible modifications

After you have selected multiple objects, you edit them:

- Shift using the mouse
  - To change the absolute position of the marked objects, position the mouse pointer over an object, and shift the multiple selection with the mouse button pressed.
  - To resize all the objects by the same ratio, grab the resizing handles of the reference object.
- Move over the work area with the icons of the toolbar
  - Change the position of the marked objects with respect to each other
  - Align the height and width of the marked objects
- Moving with the shortcut menu commands of the work area
  - Change the position of the marked objects with respect to each other
  - Align the height and width of the marked objects

See also

Overview of objects (Page 1983)

10.1.2.16 External graphics

Introduction

You can use graphics created with an external graphic program in WinCC. To use these graphics you store them in the graphic browser of the WinCC project.

You can save graphics in the graphic browser:

- When you drag-and-drop graphics objects from the "Graphics" pane into the work area, these are stored automatically in the graphic browser. The graphic names are numbered in the order of their creation, for example, "Graphic_1." Use the <F2> function key to rename the graphic.
- As a graphic file with the following formats:

  *.bmp, *.ico, *.emf, *.wmf, *.gif, *.tif, *.png, *.jpeg or *.jpg
- As an OLE object that is embedded in WinCC and is linked to an external graphic editor. In the case of an OLE link, you open the external graphic editor from WinCC. The linked object is edited using the graphic editor. An OLE link only works if the external graphic editor is installed on your PC, and supports OLE.

Use of graphics from the graphic browser

Graphics from the graphic browser are used in your screens:

- In a Graphic view
- In a graphics list
- As labeling for a button/function key

Transparent graphics

In WinCC you also use graphics with a transparent background. When a graphic with a transparent background is inserted into a graphic object of WinCC, the transparency is replaced by the background color specified for the graphic object. The selected background color is linked firmly with the graphic. If you use the graphic in another graphic object of WinCC, this object is displayed with the same background color as the graphic object that was configured first. If you want to use the graphic with different background colors, include this graphic in the graphic browser again under a different name. The additional background color is configured when the graphic is used at the corresponding graphic object of WinCC.

Managing graphics

An extensive collection of graphics, icons and symbols is installed with WinCC, for example:

In the Toolbox window of the "Graphic" pane the graphic objects are structured by topic in the "WinCC graphics folder." The link to the WinCC graphics folder cannot be removed, edited or renamed.

The "Graphics" pane is also used to manage the external graphics. The following possibilities are available:

- Creating links to graphics folders

  The external graphic objects in this folder, and in the subfolders, are displayed in the toolbox and are thus integrated in the project.
- Editing folder links
- You open the program required for editing of the external graphic in WinCC.

See also

Overview of objects (Page 1983)

10.1.2.17 Managing external graphics

Introduction

External graphics that you want to use in WinCC are managed in the "Screens" editor by using the "Tools" task card in the "Graphics" pane.

Requirement

- The "Screens" editor is open.
- The "Tools" task card is open.
- The graphics are available.
- The graphics have the following formats:

  *.bmp, *.ico, *.emf, *.wmf, *.gif, *.tif, *.jpeg or *.jpg

Creating a folder link

1. Click "My graphics folder."
2. Select "Link" in the shortcut menu.

   The "Create link to folder" dialog is opened. The dialog suggests a name for the folder link.
3. Edit the name as required. Select the path containing the graphic objects.
4. Click "OK" to confirm your input.

   The new folder link is added to the "Graphics" object group. The external graphics that are located in the target folder and in sub-folders are displayed in the toolbox.

Editing folder links

1. Select the folder link to edit.
2. Select the "Edit link..." command from the shortcut menu.
   The "Create link to folder" dialog is opened.
3. Edit the name and path of the folder link as required.
4. Click "OK" to confirm your input.

Renaming the folder link

1. Select the folder link to rename.
2. Select "Rename" from the shortcut menu.
3. Assign a name to the new folder link.

Removing a folder link

1. Select the folder link you want to delete.
2. Select "Remove" in the shortcut menu.

Edit external graphics

1. Select the graphic you want to edit.
2. Select the "Edit graphic" command from the shortcut menu.

   This opens the screen editor associated with the graphic object file.

Editing graphics folders from WinCC

1. Select the graphic you want to edit.
2. Select the "Open parent folder" command from the shortcut menu.

   The Windows Explorer opens.

See also

Overview of objects (Page 1983)

10.1.2.18 Storing an external image in the graphics library.

Introduction

To display graphics that have been created in an external graphics program in your screens, you will first have to store these graphics in the graphics browser of the WinCC project.

Requirement

- A screen has been created.
- A Graphic View is inserted in the screen.
- The Inspector window of the Graphic view is open.

To store an external graphic in the image browser:

- A graphic is available.

To store an OLE object in the browser:

- An OLE-compatible graphics program is installed on your configuration computer.

Save graphics file

1. Open the Windows Explorer.
2. Select the graphic that you want to store.
3. Drag-and-drop the graphic into the graphic browser.

Creating and saving a new graphic as an OLE object

1. Select the Graphic view on your screen.
2. In the Inspector window, select "Properties > Properties > General":
3. Open the graphic selection list.
4. Click .
5. The "Insert object" dialog box opens.

> **Note**
>
> In addition, the dialog "External application running..." will open. The dialog will not close until you exit the external application.

6. From the "Insert object" dialog box, select "New" and an object type. The settings in "Settings > "OLE settings" determine which object types are shown.
7. Click "OK." The associated graphic program is opened.

   When you are finished creating graphics, end the graphic programming software with "File > Close" or "File > Close & return to WinCC."

   The graphic will be stored in the graphic programming software standard format and added to the graphic browser.

Inserting created graphics in WinCC

Note

A new graphic object created as OLE object may not be directly accepted in WinCC when you save it to an external graphics program.

1. Reopen the dialog for inserting a graphic.
2. From the "Insert object" dialog box, select "Create from file."
3. Click the "Browse" button.
4. Navigate to the created graphic and select it.

Saving an existing graphic object as an OLE object

1. In the Inspector window, select "Properties > Properties > General":
2. Open the graphic selection list.
3. Click .
4. The "Insert object" dialog box opens.

   Note

   In addition, the dialog "External application running..." will open. The dialog will not close until you exit the external application.

5. From the "Insert object" dialog box, select "Create from file."
6. Click the "Browse" button.
7. Use the dialog to help you navigate to the folder in which the graphic file is saved.

   Note

   To import graphics files, note the following size restrictions:

   *.bmp, *.tif, *.emf, *.wmf ≤4 MB

   *.jpg, *.jpeg, *.ico, *.gif "*≤1 MB

Result

The image file is now stored in your image browser. It is shown in a screen with a Graphic view , or is added as a list element in an image list.

You can double-click OLE objects in your library to open them for editing in the corresponding graphic editor. When you have finished editing graphics, end the graphic programming software with "File > Close" or "File > Close & return to WinCC." The changes are applied to WinCC.

See also

Overview of objects (Page 1983)

10.1.2.19 Working with object groups

Basics on groups

Introduction

Groups are several objects that are grouped together with the "Group" function. You edit a group in the same way as any other object.

Overview

WinCC offers the following methods for editing multiple objects:

- Multiple selection
- Creating object groups

Editing mode

To edit an individual object in a group, select the object in the "Layout > Layers" task card.

Alternatively select "Group > Edit group" in the object group's shortcut menu.

Hierarchical groups

You add further objects or groups to extend a group. The group is enlarged by the new objects and is structured hierarchically in main and sub-groups. Such hierarchical groups must be broken up in stages. You also break up the group in the same order in which you grouped the objects or groups. It takes exactly the same number of steps to break up these hierarchical groups as it did to create them.

Rectangle surrounding the object

Only one surrounding rectangle is now displayed for the whole group. The surrounding rectangles of all objects are displayed for a multiple selection on the other hand.

Layers

All objects of a group are located in the same layer.

See also

Overview of objects (Page 1983)

Creating object groups

Introduction

The "Group" command combines multiple objects to form a group.

You can change the size and position of the group. The following rules apply:

- The system automatically adapts the position coordinates of the grouped objects when you reposition the group. The relative position of the grouped objects to the group is not affected.
- The system automatically adapts the height and width of the grouped objects in proportion to a change of the group size.
- To change the size of the group proportionately, hold down the <Shift> key and drag the rectangle around the object until has the required size.

Note

To create a hierarchical group, organize the individual groups like objects.

Requirement

- You have opened a screen which contains at least two objects.

Creating object groups

1. Select all the objects you want to organize in a group.
2. Select the command "Group > Group" from the shortcut menu.

The objects of the group are displayed with a rectangle around the objects.

Grouping objects within a group

1. Select the group you want to edit.
2. Select the command "Group > Edit group" from the shortcut menu.

   The group that you are editing is highlighted by a red frame.
3. Select the objects of a group that you want to combine into a subgroup.
4. Select the command "Group > Group" from the shortcut menu.

A subgroup with the objects is created.

Adding objects into an existing group

1. Select the group to which you want to add objects.
2. Press the <Shift> key and select the object you want to add to the group.
3. Select the "Group > Add to group" command from the shortcut menu.

The object is added to this group.

Alternative procedure

You can also edit groups in the "Layout" task card. Using drag-and-drop you can also easily edit nested groups in the "Layers" pane.

Result

The selected objects are combined in a group. The multiple selection rectangle becomes the rectangle surroundings the objects in the group. The handles are shown only for the group. The group in in the active layer.

Ungroup

Introduction

Select the "Ungroup" command to split a group into the original object fractions.

Requirement

- You have opened a screen that contains a group.

Ungroup

1. Select the group.
2. Select the "Group > Ungroup" command from the shortcut menu.

Ungrouping a group within a group

1. Select the higher-level group.
2. Select the command "Group > Edit group" from the shortcut menu.

   The group that you are editing is highlighted by a red frame.
3. Select the lower-level group.
4. Select the "Group > Ungroup" command from the shortcut menu.

Result

The lower-level group is ungrouped. The objects are assigned to the next higher group.

Alternative procedure

You can also edit groups in the "Layout" task card. Using drag-and-drop you can also easily edit nested groups in the "Layers" pane.

Adding objects to a group

Introduction

The "Add objects to group" command is used to add objects to a group, without ungrouping it first.

Requirements

A screen with one group and at least one other object is opened.

Procedure

1. Select the group.
2. Press the <Shift> key and select the object you want to add to the group.
3. Select the "Group > Add to group" command from the shortcut menu.

Result

The group consists of the original objects, and the newly-added objects. The added objects are arranged at the front of the group.

Alternative procedure

You can also edit groups in the "Layout" task card. Using drag&drop you can also easily edit hierarchical groups in the "Layers" palette.

Removing Objects from the Group

Introduction

You use the "Remove objects from group" command to remove individual objects from a group, without ungrouping it first.

You do not have to remove the object from the group to edit an object in a group. You can edit the objects of a group individually.

Requirement

- You have opened a screen that contains a group.

Removing objects from a group

To remove an object from a group:

1. Select the group.
2. Select the command "Group > Edit group" from the shortcut menu.

   The group you want to edit is highlighted by a red frame.
3. Select all objects in the group that you want to remove from the group.
4. Select the "Group > Remove from group" command from the shortcut menu.

The objects are removed from the group.

Note

If there are only two objects in the group, the menu command "Remove from group" is not available.

Deleting objects from a group

To remove an object from the group, and from the screen:

1. Select the group.
2. Select the command "Group > Edit group" from the shortcut menu.

   The group you want to edit is highlighted by a red frame.
3. Select the objects in the group that you want to delete.
4. Select "Delete" from the shortcut menu.

   Note

   If there are only two objects in the group, the menu command "Delete" is not available.

Alternative procedure

You can also edit groups in the "Layout" task card. Using drag-and-drop you can also easily edit nested groups in the "Layers" pane.

Editing an Object in a Group

Introduction

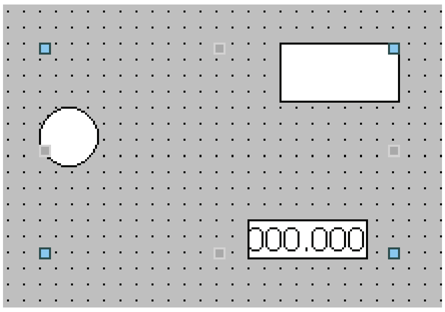
You can edit the objects of a group individually.

Requirement

You have opened a screen that contains a group.

Editing grouped objects

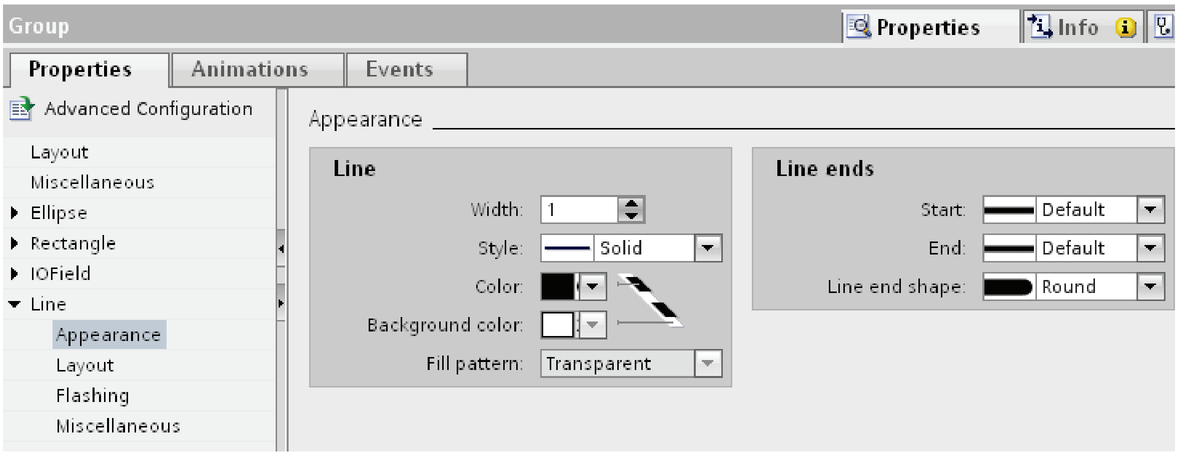
1. Select the group.



The properties of the group are displayed in the Inspector window.
2. Change the position and size of the grouped objects in "Properties > Properties >Layout."
3. Change the name of the group in "Properties > Properties > Miscellaneous."

Modifying the properties of an object within a group

1. Select the group.
2. Select the object whose properties you want to change in the Inspector window.



The properties of the object are displayed.

3. Change the object properties.

Result

Although you have edited the object, it is still an element of the group. These changes do not affect the other objects of the group.

10.1.2.20 Configuring the keyboard access

Overview of keyboard access

Introduction

For keyboard units without a mouse, the operator activates control objects using the <Tab> key. You can set up keyboard input to make the process easier to run, and to make sure that the operator enters all the necessary values. If you are using the keyboard, use the <Tab> key to activate the objects in a certain order, and to enter the necessary values.

For HMI devices without key, you can simulate the <Tab> key by configuring the "SimulateSystemKey" system function to a function key.

Operator authorizations and operator control enables

If you configure an object for operator input with the <Tab> key, the object must have both an operator authorization, and an operator control enable.

Editing the tab sequence

The tab sequence is determined automatically when the control objects are created. The numbers of the tab sequence are assigned in the sequence in which the libraries are created.

It makes sense to change the tab sequence in the following cases:

- The operator changes directly to a specific control object.
- The screen requires a specific sequence

Change to the tab sequence mode to change the tab sequence. In this mode, the tab sequence number is displayed at the top left of the control objects. The tab sequence numbers of hidden objects are also shown. The distribution of these numbers is edited using the mouse.

Note

Other functions are not available in the tab sequence mode.

See also

Example: Inserting and configuring a rectangle (Page 2019)

Overview of objects (Page 1983)

Defining the Operator Authorization and Operator Control Enable for an Object

Introduction

If you configure an object for operator input with the <Tab> key, the object must have both an operator authorization, and an operator control enable.

Requirement

You have opened a screen which contains at least one object.

Procedure

1. Select the object.
2. Select "Properties > Properties > Security" in the Inspector window.
3. Select the operator authorization under "Authorization."
4. Activate the authorization to operate.

Result

The operator can use the <Tab> key in Runtime to select the object.

See also

Example: Inserting and configuring a rectangle (Page 2019)

Setting the tab sequence order

Introduction

All operable objects can be reached in runtime with the <Tab> key. You use the "Tab sequence" command to define the order in which the operator can activate objects in runtime.

Note

You cannot reach objects with the "Output" or "Two states" mode in runtime with the <Tab> key.

You can operate the screen in runtime:

- Using the <Tab> key
- Using the mouse
- Using a configured hotkey

Requirement

- The active screen contains operable objects.
- No object is selected.
- The objects have been enabled for use in runtime, and have operator authorization.

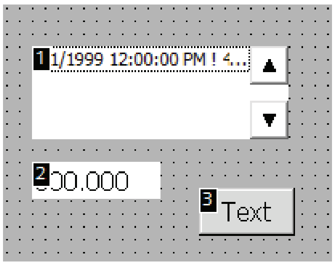
Procedure

1. Select "Edit tab sequence" in the "Edit" menu.

   Tab sequence mode is activated. The tab sequence number is displayed for all objects that can be used. The tab sequence number is also displayed for hidden objects.

2. Use edit the tab sequence mode, click the accessible objects in the order in which you want them to be activated using <Tab> in runtime.

   The following figure shows how the tab sequence is defined in the screen. In runtime, the <Tab> key first activates the alarm view (number 1), then the I/O field (number 2), and then the button (number 3):

3. To exclude a screen object from the tab sequence, press the key combination <Shift+Ctrl> and click on the desired object.

   The tab sequence number is no longer displayed in the screen object. The screen object is now excluded from the tab sequence. The remaining tab sequence numbers are automatically decreased by 1.

4. To reenter an excluded screen object in the tab sequence, repeat step 3.

   The screen object entered as the first object in the tab sequence.

Result

The operator selects the objects in the specified order in Runtime with the <Tab> key.

See also

Example: Inserting and configuring a rectangle (Page 2019)

10.1.2.21 Examples

Example: Inserting and configuring a rectangle

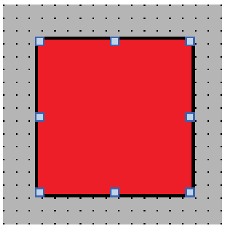
Task

In this example, you insert a rectangle in a screen. You can configure the following properties:

- Name = "MyRectangle"
- Position = (20, 20)
- Size = (100,100)
- Color = red
- Black frame 2 pixels wide

Principle

The rectangle is a closed object which can be filled with a color or pattern. The height and width of a rectangle can be adjusted to allow its horizontal and vertical alignment.



Overview

Carry out the following steps in order to create a rectangle:

- Inserting a rectangle
- Configuring a rectangle

See also

Basics on groups (Page 2009)

Overview of objects (Page 1983)

Example: Inserting a rectangle

Task

In this example, you insert and rename a rectangle. Do not use the special characters ?, ", /, \, *, <, > for the name.

Requirement

- A screen is open.
- The Inspector window is open.
- The "Tools" task card is open.

Procedure

1. Click the "Basic objects" palette in the "Tools" task card.
2. Drag the "Rectangle" object into the screen.
3. In the Inspector window, select "Properties > Properties > Miscellaneous".
4. Type in the new name "MyRectangle".

Result

The rectangle is now inserted and named "MyRectangle". The rectangle has the default properties of the "rectangle" object.

Example: Configuring a rectangle

Task

In this example you configure a rectangle:

- Color = red
- Black frame 2 pixels wide
- Position = (20, 20)
- Size = (100,100)

Changing the color of the rectangle

To change the color of the rectangle:

1. Select the rectangle.
2. Define the background color in "Properties > Properties > Appearance > Background > Color" in the Inspector window.
3. Select "Solid" as the fill pattern.
4. Define the color for the border in "Properties > Properties > Appearance > Border > Color" in the Inspector window.
5. Enter the value "2" for "width".
6. Select "Solid" as the "Style".

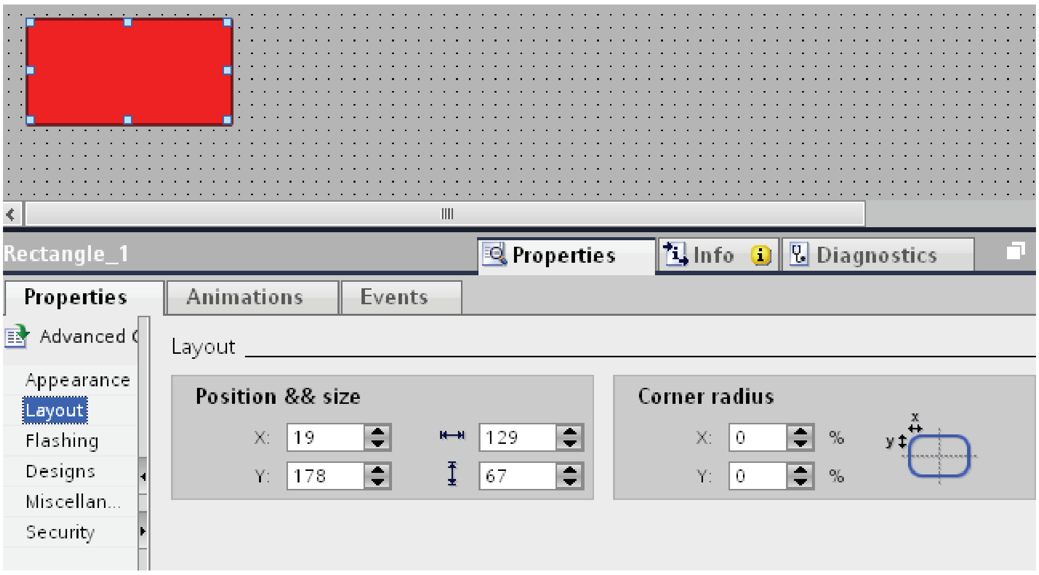
Interim result

The rectangle is red and has a black frame with a width of two pixels.

Repositioning and resizing the rectangle

To change the position and size of the rectangle:

1. Select the rectangle.
2. In the Inspector window, select "Properties > Properties > Layout".



Rectangle_1

Layout

Position && size
X: 19
Y: 178
129
67

Corner radius
X: 0 %
Y: 0 %

3. Set "20" for the both the X and Y coordinates under "Position & Size".
4. Set "100" for the height and for the width.

Result

The rectangle is positioned at the coordinates (20, 20), and has a width and height of 100 pixels.

10.1.3 Working with text lists and graphics lists

10.1.3.1 Working with text lists

Basics on text lists

Introduction

Texts are assigned to the values of a tag in a text list. Assign the text list to a symbolic I/O field for example in the configuration. This supplies the text to be displayed to the object. The text lists are created in the ""Text List" editor. You configure the interface between the text list and a tag at the object that uses the text list.

The selection of objects that can have a text list assigned depends on the Runtime.

Application

The text list is used, for example, to display a drop-down list in a symbolic I/O field.

If the symbolic I/O field is a display field, the associated texts will differ according to the value of the configured tags. If the symbolic I/O field is an input field, the configured tag assumes the associated value when the operator selects the corresponding text in Runtime.

Note

Display of tag values without text

The display of tag values to which no text has been assigned depends on the Runtime:

- The display and operating object remains empty.
- Three asterisks *** are displayed.

Multilingual texts

You can configure multiple languages for the texts in a text list. The texts will then be displayed in the set language in Runtime. To this purpose you set the languages in the Project window under "Language support > Project languages."

Configuration steps

The following steps are necessary to display texts in a symbolic I/O field for example:

1. Creating the text list
2. Assignment of the texts to values or value ranges of a text list
3. Assignment of a text list in the display object, e.g. the symbolic I/O field

Creating a text list

Introduction

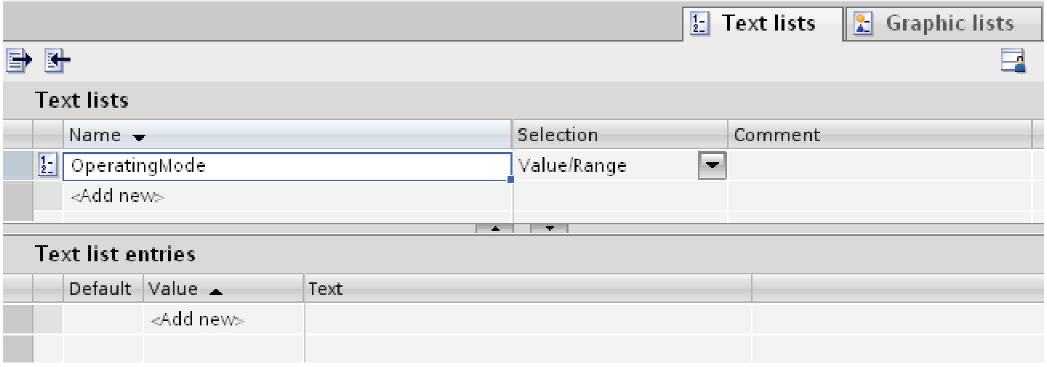
The text list allows you to assign specific texts to values and to output these in Runtime, for example in a symbolic I/O field. The type of symbolic I/O field can be specified, for example as a pure input field.

The following types of list are available:

- Value/Range
- Bit
- Bit Number

Procedure

1. Double-click "Text and graphics lists" in the project window.
2. Open the "Text lists" tab.



Text lists

Name	Selection	Comment
OperatingMode	Value/Range	
<Add new>		

Text list entries

Default	Value	Text
	<Add new>	

3. Click "Add" in the "Text lists" table.

   The Inspector window of the text list is open.
4. Assign a name to the text list that indicates its function.
5. Select the text list type under "Selection":
   – Value/Range: Text from the text list is displayed when the tag has a value that lies within the specified range.
   – Bit (0,1): A text from the text list is displayed when the tag has the value 0. A different text from the text list is displayed when the tag has the value 1.
   – Bit number (0-31): Text from the text list is displayed when the tag has the value of the assigned bit number.
6. Enter a comment for the text list.

Note

You must not use semicolons in the texts in a text list. The semicolon is a control character and is automatically deleted from a text.

Result

A text list is created.

Assigning texts and values to a range text list

Introduction

For each area text list you specify which texts are displayed at which value range.

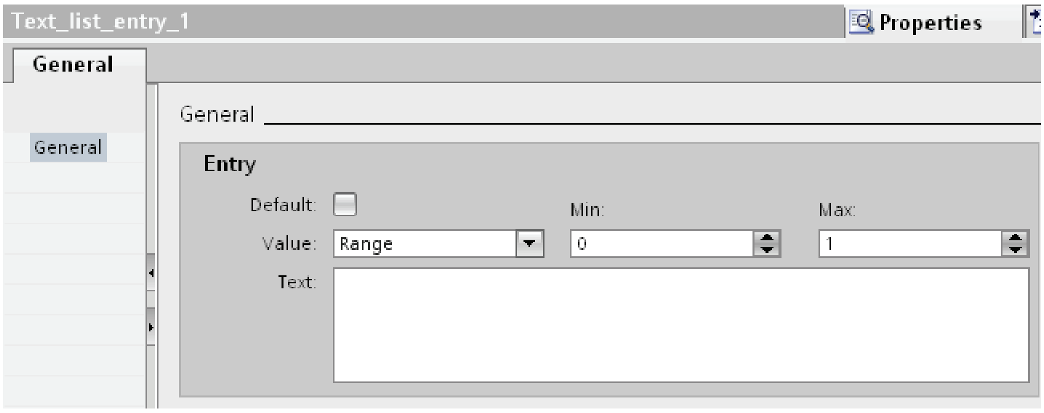
Requirement

- The "Text and graphics list" editor is open.
- The "Text lists" tab is open.
- An area text list has been created and selected.

Procedure

1. Click "Add" in the "Text list entries" table.

   The Inspector window for this list entry opens.



Text_list_entry_1

General

Entry

Default: □

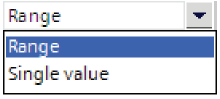
Value: Range

Min: 0

Max: 1

Text:

2. Select the setting "Range" in "Properties > Properties > General > Value" in the Inspector window.



   – Enter the value "1" for "Min" for example.

   – Enter the value "20" for "Max" for example.

   – For "Text", enter the text that is displayed in Runtime if the tag is within the specified value range.

Note

Use a maximum of 255 characters and no semicolons for the text.

3. Click "Add" in the "Text list entries" table. A second list entry is created.
4. Select the setting "Range" in "Properties > Properties > General > Value" in the Inspector window.
   - Enter the value "21" for "Min" for example.
   - Enter the value "40" for "Max" for example.
   - For "Text", enter the text that is displayed in Runtime when the tag is within the specified value range.
5. If required, activate the "default entry".

   The entered text is always displayed when the tag has an undefined value. Only one default entry is possible per list.

Result

An area text list is created. Texts have been assigned to the possible value ranges.

Assigning texts and values to a bit text list

Introduction

For each text list, you specify which text is displayed at which bit value.

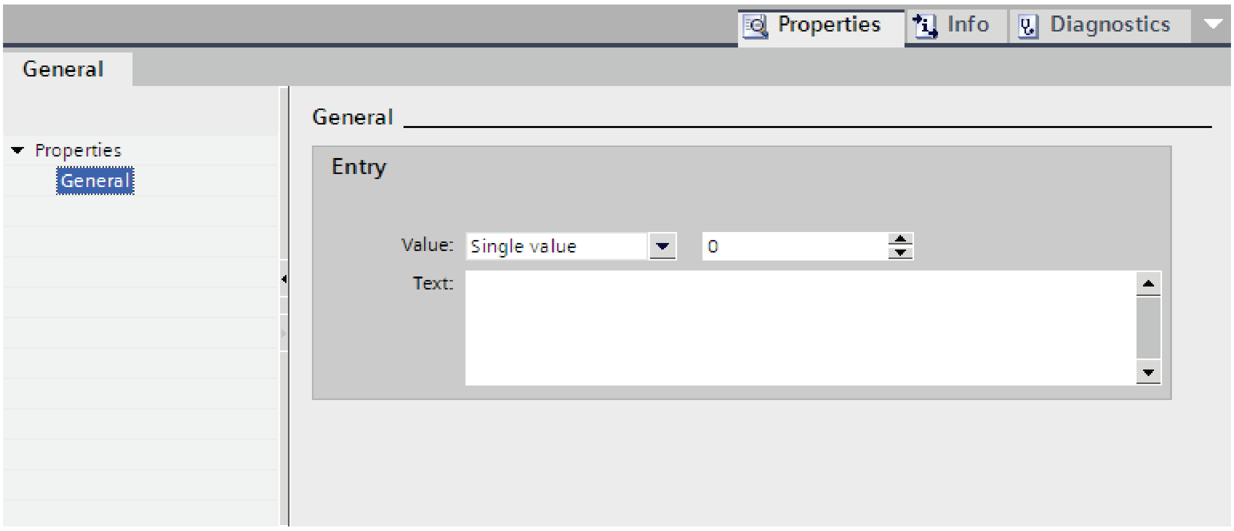
Requirement

- The "Text and graphics list" editor is open.
- The "Text lists" tab is open.
- A bit text list has been created and selected.

Procedure

1. Click "Add" in the "Text list entries" table.

The Inspector window for this list entry opens.



Value: Single value 0

Text:

2. Select the setting "Single value" in "Properties > Properties > General > Value" in the Inspector window.
   - Enter "0" for "Value."
   - Enter the text which is displayed in Runtime under "Text" if the bit tag is set to "0".

   Note

   Use a maximum of 255 characters and no semicolons for the text.

3. Click "Add" in the "Text list entries" table. A second list entry is created.
4. Select the setting "Single value" in "Properties > Properties > General > Value" in the Inspector window.
   - Enter "1" under "Value."
   - Enter the text which is to be displayed in Runtime under "Text" if the bit tag is set to "1".

Result

A bit text list is created. Texts that appear in Runtime are assigned to the possible values "0" and "1".

Assigning texts and values to a bit number text list

Introduction

For each bit number text list you specify which texts are displayed at which bit number.

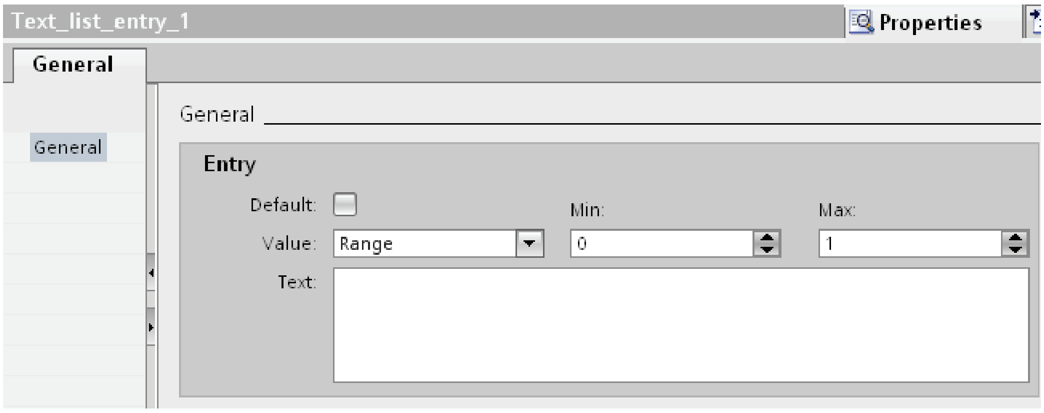
Requirement

- The "Text and graphics list" editor is open.
- The "Text lists" tab is open.
- A bit number text list has been created and selected.

Procedure

1. Click "Add" in the "Text list entries" table.

   The Inspector window for this list entry opens.



Text_list_entry_1

Default: □

Value: Range

Min: 0

Max: 1

Text:

2. Select the setting "Single value" in "Properties > Properties > General > Value" in the Inspector window.
   - Enter "10", for example, for "Value".
   - Under "Text", enter the text that is displayed in Runtime when the tag has the value "10".

   Note

   Use a maximum of 255 characters and no semicolons for the text.

3. If required, activate the "default entry".

   The entered text is always displayed when the tag has an undefined value. Only one default entry is possible per list.
4. Create further list entries for additional bit numbers of the same text list.

Result

A bit number text list is created. Texts that appear in Runtime are assigned to the specified bit numbers.

Configuring objects with a text list

Introduction

The output value and value application for text lists are specified in the display and operating object that displays the texts of the text list in Runtime. The properties of these objects are configured as required.

Requirement

- A text list is created.
- You have created a tag.
- The "Screens" editor is open.
- A screen with a symbolic I/O field is open. The object is edited.

Procedure

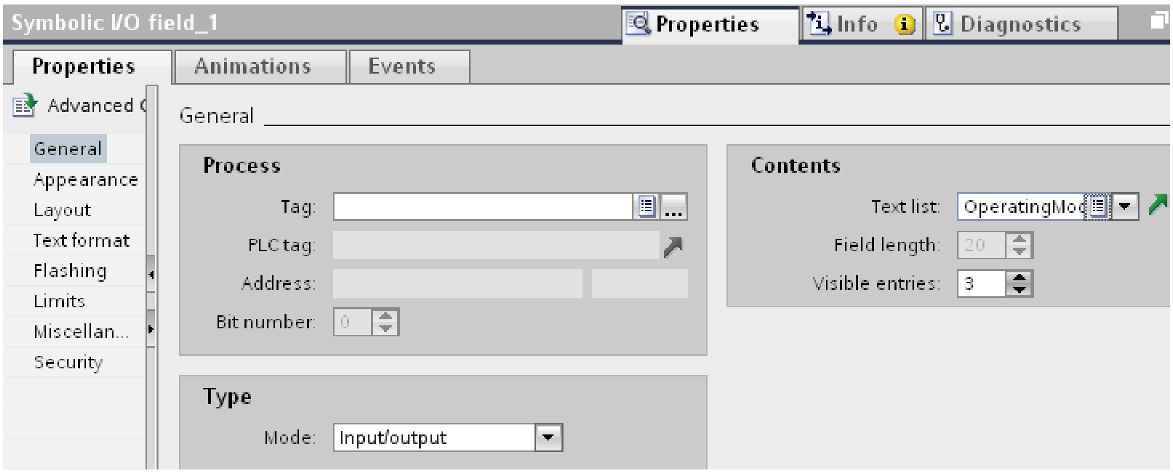
1. Select the text list which you want to have displayed in Runtime in "Properties > Properties > General > Text list" in the Inspection window.
2. Select the setting "Output" as the "Mode".

   Note

   Runtime dependency

   Different field types are available for a symbolic I/O field depending on the Runtime.

3. Select the tag the value of which determines the display in the symbolic I/O field as "Tag".



Tag:
PLC tag:
Address:
Bit number: 0
Text list: OperatingMod
Field length: 20
Visible entries: 3
Mode: Input/output

Result

The defined texts of the text list are displayed in the symbolic I/O field in Runtime when the tag has the specified value.

10.1.3.2 Working with graphics lists

Basic principles of graphics lists

Introduction

The possible values of a tag are assigned to specific graphics in a graphics list. During configuration, you can create a graphics list for a button or a graphic I/O field. This supplies the graphics to be displayed to the object.

The graphics lists are created with the "Text and graphics list" editor. You configure the interface between the graphics list and a tag at the object that uses the graphics list. The availability of the graphics list is determined by the HMI device used.

Application

You can configure the graphics list for the following situations:

- Drop-down list with a graphic I/O field
- State-specific graphic for a button

The graphics in a graphics list can be configured as multilingual. The graphics will then be displayed in the set language in runtime.

Graphic sources

Graphics can be added to the graphics list from the following sources:

- By selecting from a graphic browser
- By selecting an existing file
  You can use the following file types:
  *.bmp, *.ico, *.emf, *.wmf, *.gif, *.tiff, *.png, *.jpeg and *.jpg.
- By creating a new file

Function

If the graphic I/O field is a display field, the associated graphics will differ according to the value of the configured tags. If the graphic I/O field is an input field, the configured tag assumes the associated value when the operator selects a graphic in runtime.

Configuration steps

The following tasks are required to display graphics, for example, in a graphic I/O field:

1. Creating the graphics list
2. Assignment of the graphics to values or value ranges of a graphics list
3. Assigning a graphics list in the display object, for example the graphic I/O field

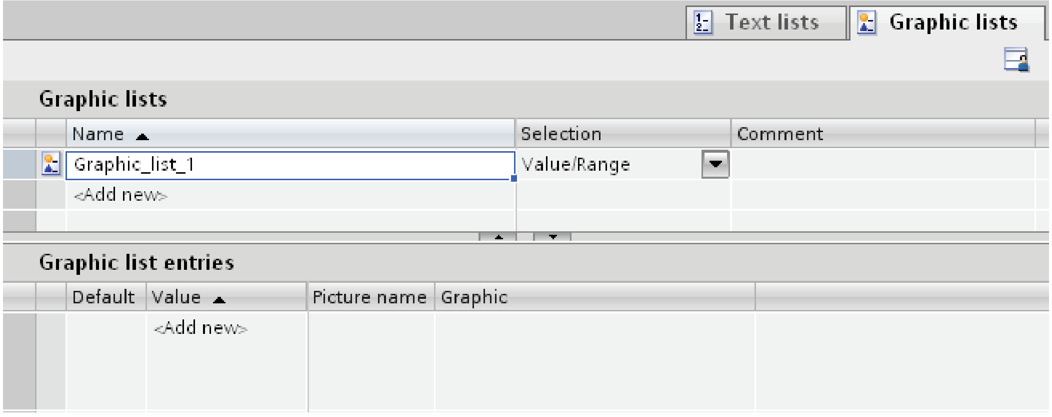
Creating a graphics list

Introduction

The graphics list allows you to assign specific graphics to variable values and to output these in a graphic IO field in Runtime. You can specify the type of graphic I/O field, for example as a pure output field.

Procedure

1. Double-click "Text and graphics lists" in the project window.
2. Open the "Graphics lists" tab.



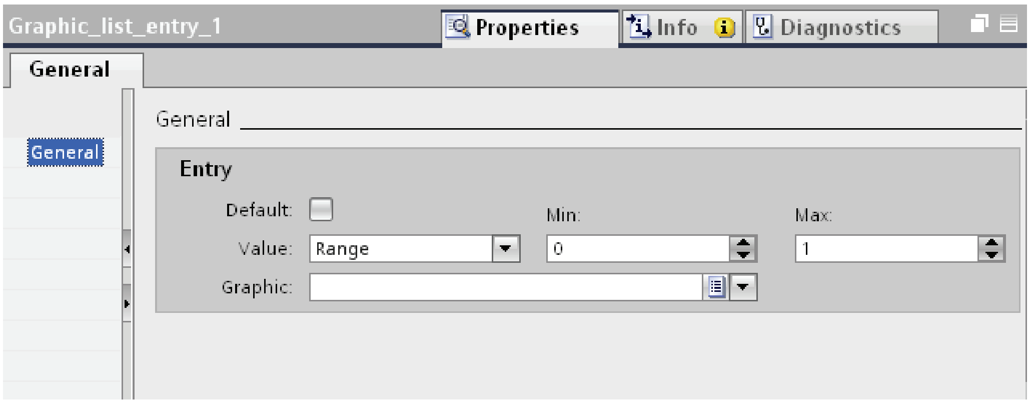
Graphic lists

Name	Selection	Comment
Graphic_list_1	Value/Range	
<Add new>		

Graphic list entries

Default	Value	Picture name	Graphic
	<Add new>		

3. Click "Add" in the "Graphics lists" table. The Inspector window of the graphics list will open up.



Entry

Default: □
Value: Range
Min: 0
Max: 1
Graphic:

4. Assign a name to the graphics list that indicates its function.
5. Select the graphics list type "Bit number (0 - 31)" for example under "Select".
6. Enter a comment for the graphics list.

Result

A graphics list of the type "Range (0 - 31)" is created.

Assigning a graphic and values to a range graphics list

Introduction

For each area graphics list you specify which graphics are displayed at which value range.

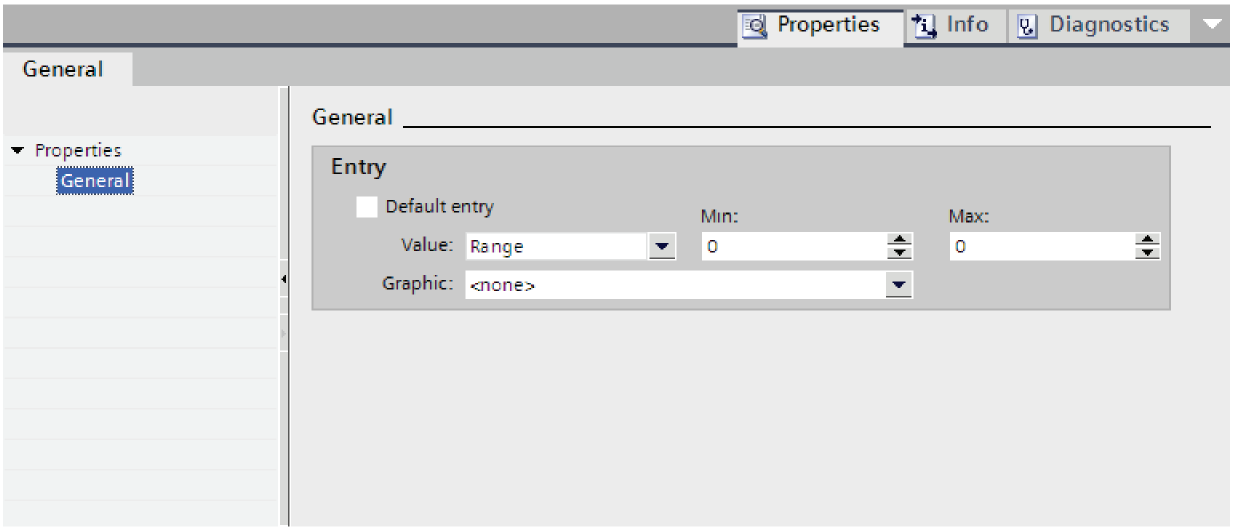
Requirement

- The "Text and graphics list" editor is open.
- The "Graphics list" tab is open.
- An area graphics list has been created and selected.

Procedure

1. Click "Add" in the "Graphics list entries" table.

   The Inspector window for this list entry opens.



Entry

□ Default entry

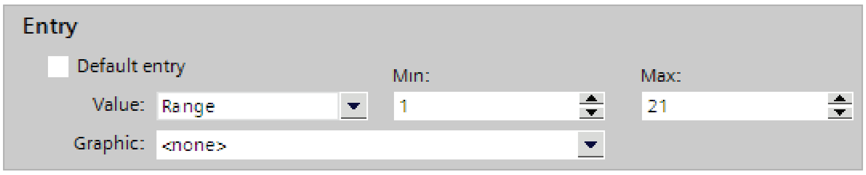
Value: Range

Min: 0

Max: 0

Graphic: <none>

2. Select the settings "Range" in "Properties > Properties > General > Value" in the Inspector window:
   - Enter the value "1" for "Min" for example.
   - Enter the value "20" for "Max" for example.
   - Select a graphic that is displayed in Runtime when the tag is within the specified value range.



Entry

□ Default entry

Value: Range

Min: 1

Max: 21

Graphic: <none>

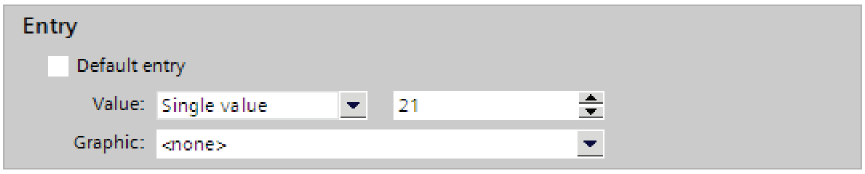
Note

As an alternative to the drop-down menu, you can insert graphics from libraries or from your file system:

1. Select a graphic in the library or in your file system.
2. Drag-and-drop the graphic into the "Graphics list entries > Graphic" table.

3. Click "Add" in the "Graphics list entries" table. A further list entry is created.

4. Select the settings "Single value" in "Properties > Properties > General > Value" in the Inspector window:
   - Enter the value "21" for example.
   - Select a graphic which is displayed in Runtime if the bit "21" is set in the tag.



Entry

□ Default entry

Value: Single value 21

Graphic: <none>

5. If required, activate the "default entry".

   The graphic is always displayed when the tag has an undefined value. Only one default entry is possible per list.

Result

An area graphics list is created. Graphics that appear in Runtime are assigned to the possible values.

Assigning graphics and values to a bit graphics list

Introduction

For each graphics list you specify which graphic is displayed at which bit value.

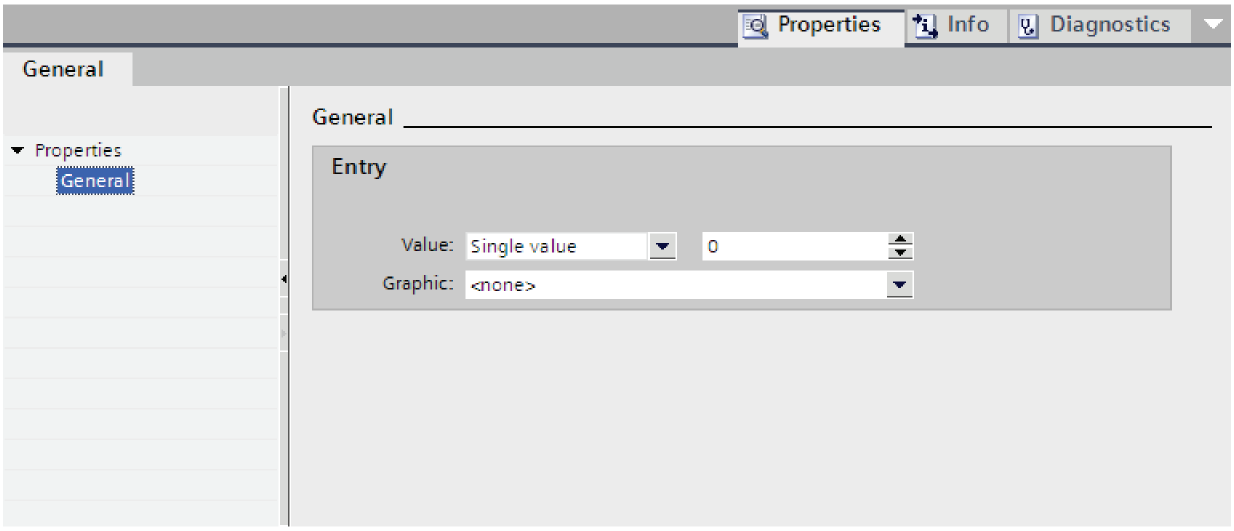
Requirement

- The "Text and graphics list" editor is open.
- The "Graphics list" tab is opened.
- A bit graphics list has been created and selected.

Procedure

1. Click "Add" in the "Graphics list entries" table.

   The Inspector window for this list entry opens.



Value: Single value 0

Graphic: <none>

2. Select the settings "Single value" in the inspector window "Properties > Properties > General > Value":
   - Enter "0" as the value.
   - Select a graphic which is displayed in Runtime if the bit "0" is set in the tag.

   Note

   As an alternative to the drop-down menu, you can insert graphics from libraries or from your file system:

   1. Select a graphic in the library or in your file system.
   2. Drag-and-drop the graphic into the "Graphics list entries > Graphic" table.

3. Click "Add" in the "Graphics list entries" table. A new list entry is created.
4. Select "Properties > Properties > General > Value > Single value": in the Inspector window.
   - Enter "1" as the value.
   - Select a graphic which is displayed in Runtime if the bit "1" is set in the tag.

Result

A bit graphics list is created. Graphics that appear in Runtime are assigned to the values "0" and "1".

Assigning graphics and values to a bit number graphics list

Introduction

For each bit number graphics list you specify which graphics are displayed at which bit number.

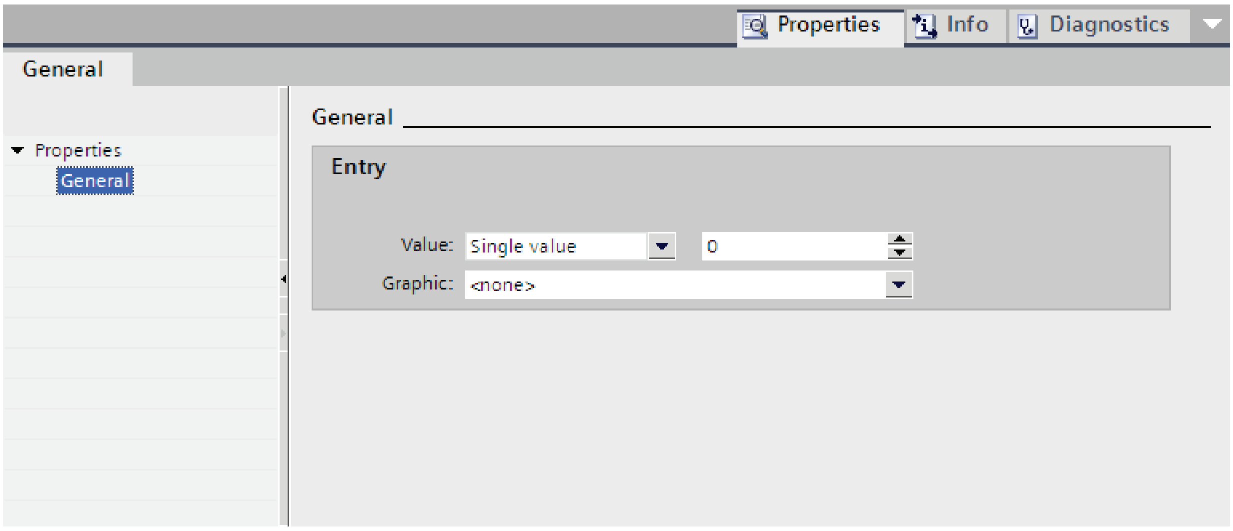
Requirement

- The "Text and graphics list" editor is open.
- The "Graphics list" tab is open.
- A bit number graphics list has been created and selected.

Procedure

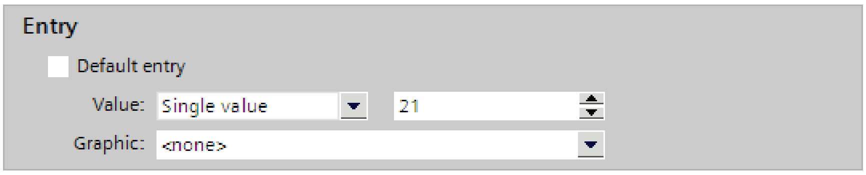
1. Click "Add" in the "Graphics list entries" table.

   The Inspector window for this list entry opens.



Value: Single value 0
Graphic: <none>

2. Select the settings "Single value" in "Properties > Properties > General > Value" in the Inspector window:
   - Enter the value "0" for example.
   - Select a graphic which is displayed in Runtime if the bit "0" is set in the tag.



□ Default entry
Value: Single value 21
Graphic: <none>

Note

As an alternative to the drop-down menu, you can insert graphics from libraries or from your file system:

1. Select a graphic in the library or in your file system.
2. Drag-and-drop the graphic into the "Graphics list entries > Graphic" table.

3. If required, activate the "default entry".

   The graphic is always displayed when the tag has an undefined value. Only one default entry is possible per list.
4. Create further list entries for additional bit numbers of the same graphics list.

Result

A bit number graphics list is created. Graphics that appear in Runtime are assigned to the specified bit numbers.

Configuring objects with a graphics list

Introduction

The output value and value application for graphics list are specified in the display and operating object that displays the graphics of the graphics list in Runtime. The properties of these objects are configured as required.

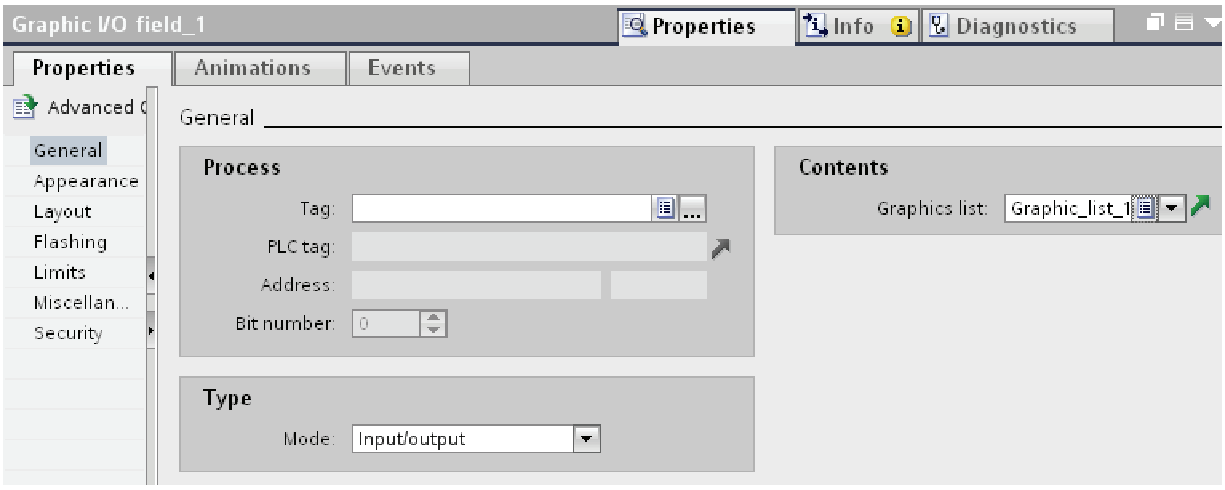
Requirement

- A graphics list is created. The values have been defined. Graphics have been assigned to the values.
- You have created a tag.
- The "Screens" editor is open.
- A screen with a graphics I/O field is open. The object is edited.

Procedure

1. Select the graphics list the graphics of which you want to have displayed in Runtime in "Properties > Properties > General > Graphics list" in the inspector window.

   Select the setting "Input/Output" as the "Mode".



Bit number: 0

Mode: Input/output

Graphics list: Graphic_list_1

> **Note**
>
> **Runtime dependency**
>
> Different field types are available for a graphic I/O field depending on the Runtime.

2. As "Tag", select the tag whose values are defined by the display in the graphic I/O field.

Result

The defined graphics of the graphics list are displayed in the graphic I/O field in Runtime when the tag has the specified value.

10.1.4 Dynamizing screens

10.1.4.1 Basics on dynamization

Dynamizing objects

In WinCC you dynamize objects to map your system and show processes on HMI devices.

You implement dynamizations by

- Animations
- Tags
- System functions

One example is the mapping of a tank, the liquid level of which rises or falls in relation to a process value.

The options for dynamization depend on the object involved. When you copy an object, its dynamization functions are included.

See also

Dynamization in the inspector window (Page 2038)

Configuring a new animation (Page 2040)

Basic on events (Page 2049)

10.1.4.2 Dynamization in the inspector window

Introduction

Basically, you can dynamize all the screen objects which you have configured in a screen. Which dynamization possibilities and which events are available depends on the device and the selected object.

Animations

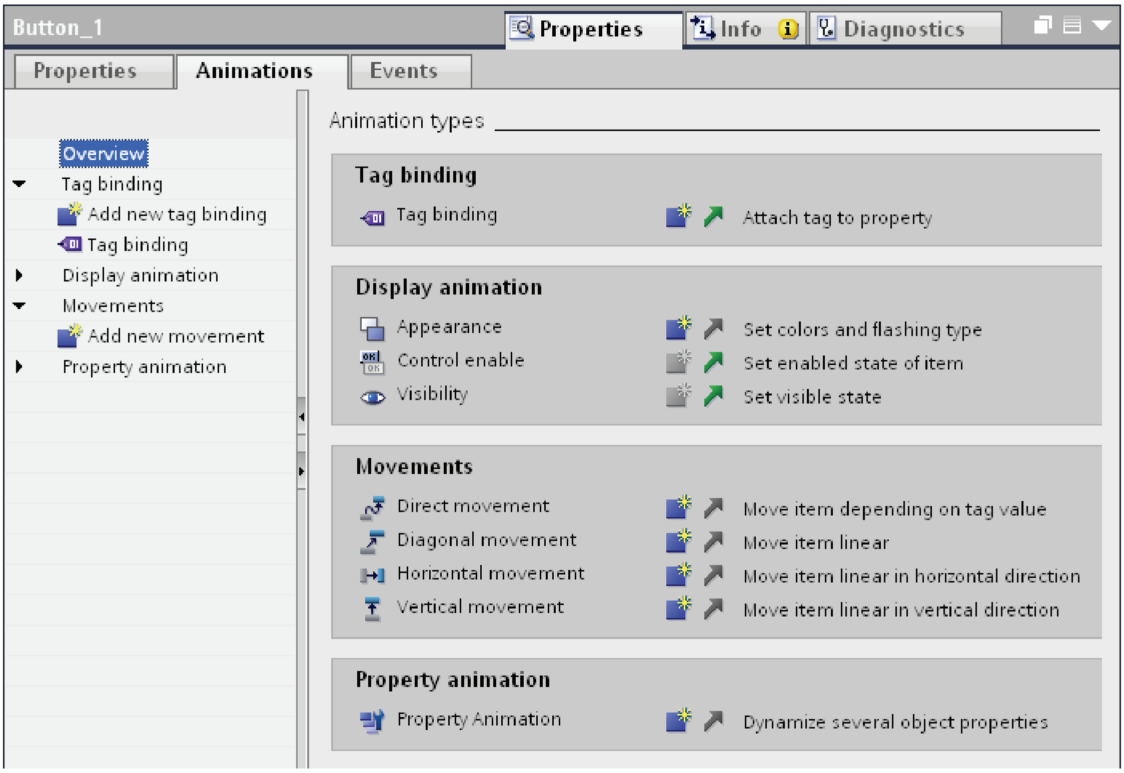
WinCC helps you to implement dynamization using predefined animations. If you want to animate an object, first configure the desired animation in the object's inspector window. Then adapt the animation to the requirements of your project.

The selection of the supported animations depends on the HMI device and the selected object. You choose between the following types of animation:

- Layout: Appearance, visibility
- Movements: direct, diagonal, horizontal and vertical movement
- Variable binding

You can configure the "Variable binding" type of animation several times for one object.

You configure animations in the "Properties > Animations" inspector window.



Tag binding

Tag binding	Attach tag to property

Display animation

Appearance	Set colors and flashing type
Control enable	Set enabled state of item
Visibility	Set visible state

Movements

Direct movement	Move item depending on tag value
Diagonal movement	Move item linear
Horizontal movement	Move item linear in horizontal direction
Vertical movement	Move item linear in vertical direction

Property animation

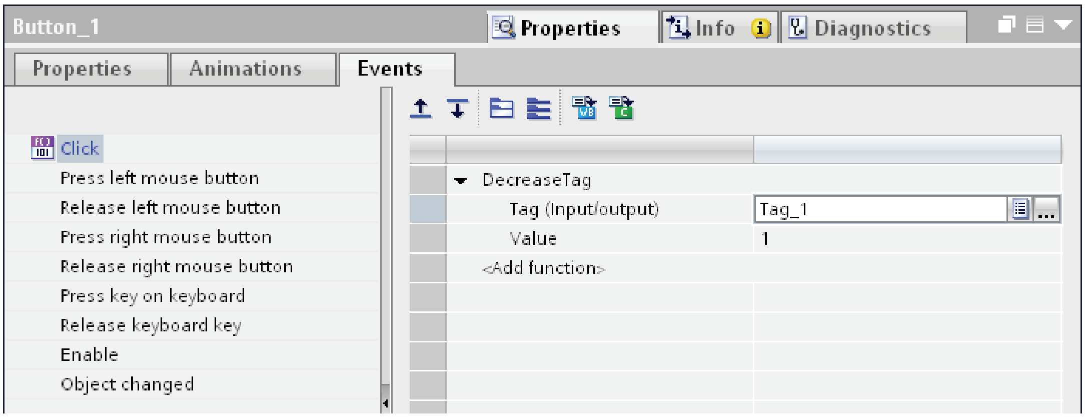
Property Animation	Dynamize several object properties

Events

Operable objects also react to events, such as a mouse click.

You configure a function list with system functions on an event. The system functions are processed as a reaction to the triggered event.

You configure events in the "Properties > Events" inspector window.



DecreaseTag	
Tag (Input/output)	Tag_1
Value	1
<Add function>	

You will find further information in "Working with function lists (Page 2287)".

See also

Basics on dynamization (Page 2038)

10.1.4.3 Dynamization with animations

Configuring a new animation

Introduction

Use predefined animations to dynamize screen objects.

Requirement

- You have opened a screen which contains at least one dynamic object.
- The Inspector window is open.
- The toolbox window is displayed.

Procedure in the inspector window

1. In the Inspector window, select "Properties > Animations".
2. Select the animation you want to use.
3. Click the button.

Procedure in the "Animations" task card.

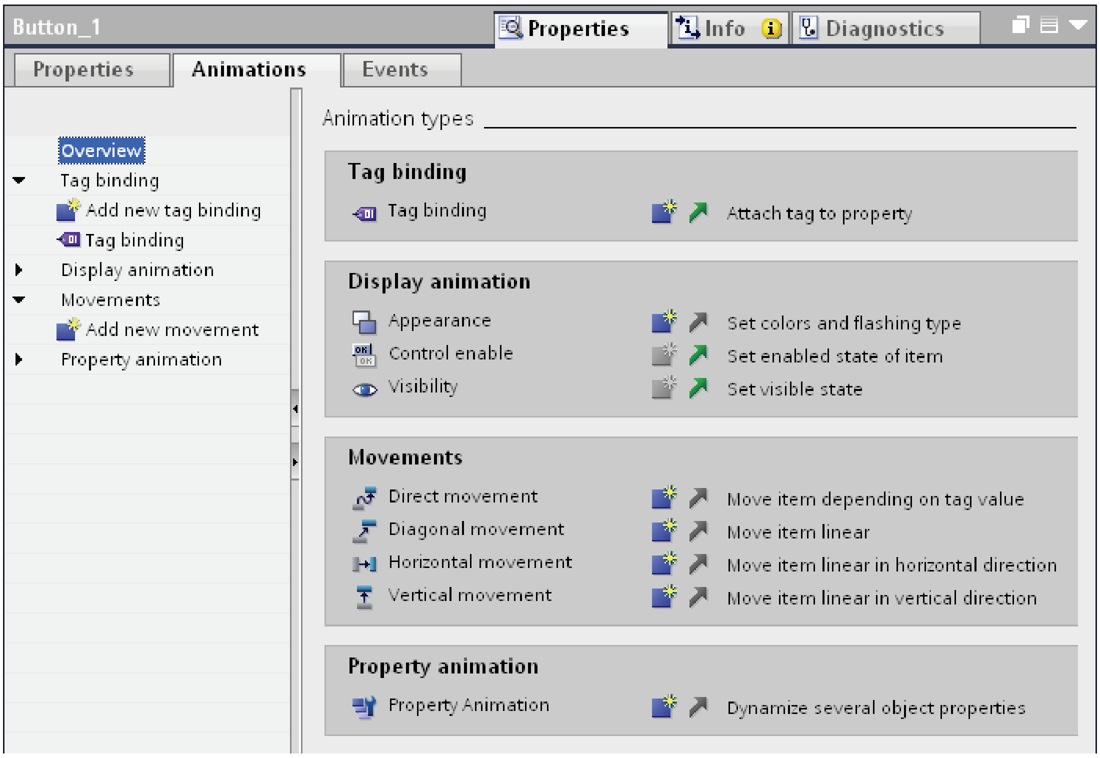
1. Open the object group containing the required animation in the "Animations" task card.
2. Drag the animation onto the object that you want to make dynamic.

Alternatively you select the object in the screen and double click the desired animation in the "Animation" task card.

Result

The animation appears in the Inspector window of the object. You configure the animation in the following steps.

In the animations overview the green arrow indicates which animation is already configured. The configured animation opens in the inspector window when you click the green arrow.



See also

Dynamizing the visibility of an object (Page 2046)

Basics on dynamization (Page 2038)

Dynamize appearance of an object

Introduction

The appearance of a screen object is controlled by changing the value of a tag in runtime. When the tag adopts a certain value, the screen object changes its color or flashing characteristics according to the configuration.

Type

Areas or single values of the tag are observed in Runtime depending on the selection. The appearance of the object changes according to the configuration.

Requirement

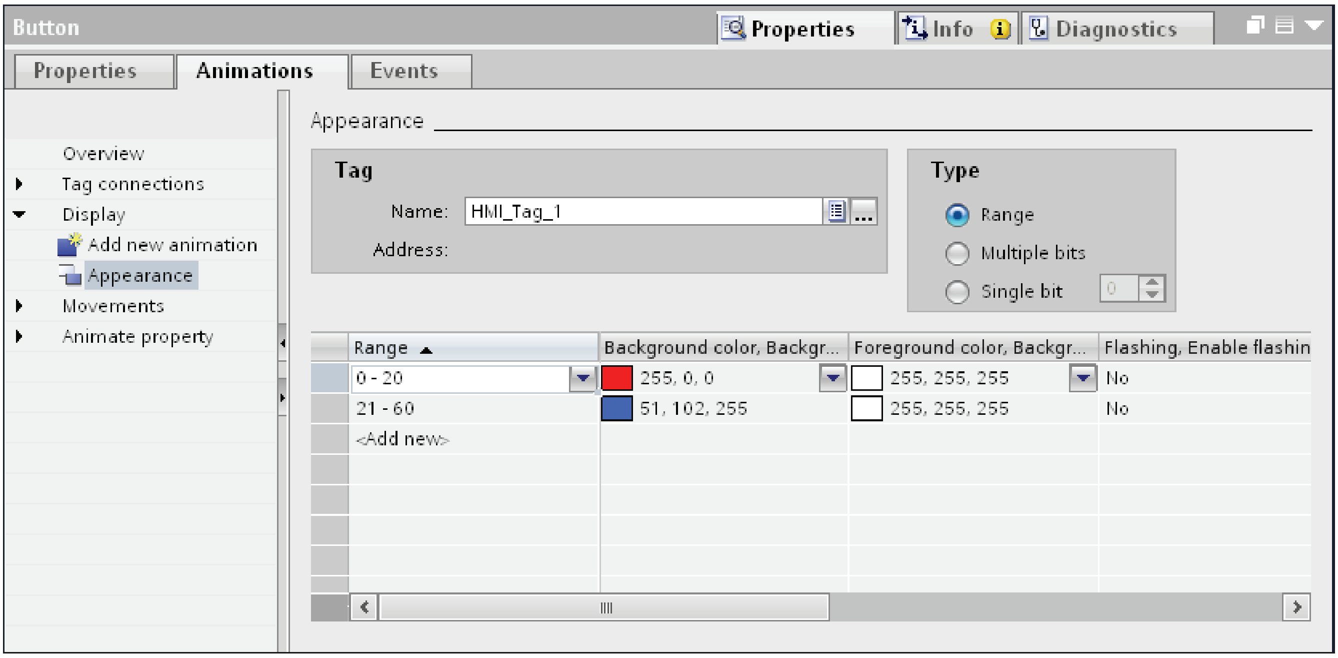
- A screen is open.
- A dynamic object is contained and selected in the screen.
- The Inspector window is open.
- The toolbox window is displayed.

Procedure

1. In the Inspector window, select "Properties > Animations".

   The animations available for the selected object are displayed.
2. Select the animation "Appearance" and click the button.

   The parameters of the animation are displayed.
3. Select a tag in "Tag > Name".
4. Select "Type > Range" for example.
5. Click "Add" in the table.
6. Enter the tag interval "0 - 20" in the "Range" column for example.
7. For "Foreground color" and "Background color", select the color the object is to acquire in Runtime when the tag reaches the interval.
8. Select a flashing mode for the object from the "Flashing" list.
9. Repeat steps 5 to 8 to create another tag interval, e.g. "21 - 60".



Appearance

Tag

Name: HMI_Tag_1

Address:

Type

● Range
○ Multiple bits
○ Single bit 0

Range	Background color, Backgr...	Foreground color, Backgr...	Flashing, Enable flashin
0 - 20	🟥 255, 0, 0	□ 255, 255, 255	No
21 - 60	🟦 51, 102, 255	□ 255, 255, 255	No
<Add new>			

Result

In Runtime, the flashing response, and color of the object change dynamically according to the process value returned in the tag.

Configuring horizontal movement

Introduction

You can configure dynamic objects in such a way that they move on a certain track. The movement is controlled via tags. The object moves every time the tag is updated.

You can only program one type of movement for each object.

Requirement

- You have created a tag.
- You have opened a screen which contains at least one dynamic object.
- The Inspector window is open.
- The toolbox window is displayed.

Procedure

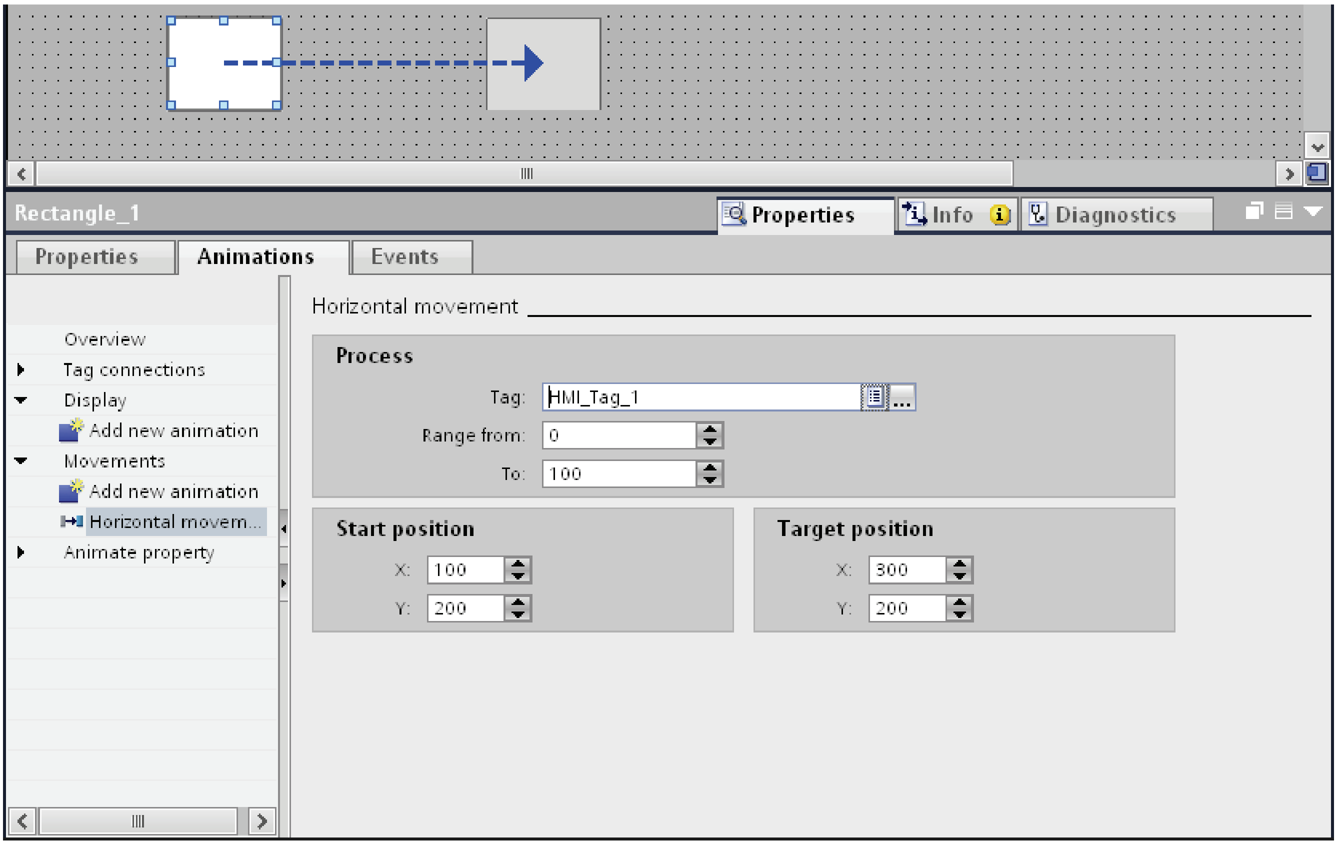
1. Select the screen object you want to control dynamically.

   The object properties are displayed in the Inspector window.
2. In the Inspector window, select "Properties > Animations".

   The animations available for the selected object are displayed.
3. Select "Horizontal movement" and click the button.

   The parameters of the animation are displayed.

   A transparent copy of the object is shown in the work area, which is connected to the source object by means of an arrow.
4. Select a tag for control of movement.
5. Move the object copy to the relevant destination. The system automatically enters the pixel values of the final position in the Inspector window.
6. Customize the range of values for the tag as required.



Horizontal movement

Process

Tag: HMI_Tag_1

Range from: 0

To: 100

Start position

X: 100

Y: 200

Target position

X: 300

Y: 200

Result

In Runtime, the object moves in response to every change in value of the tag that is used to control the movement. The direction of movement corresponds to the configured type of movement "horizontal".

Note

You configure vertical and diagonal movement in the same way as horizontal movement

Configuring direct movement

Introduction

The object is moved respectively in x direction and y direction with "Direct movement". Two tags define the number of pixels by which the object moves from its original static start position.

Requirement

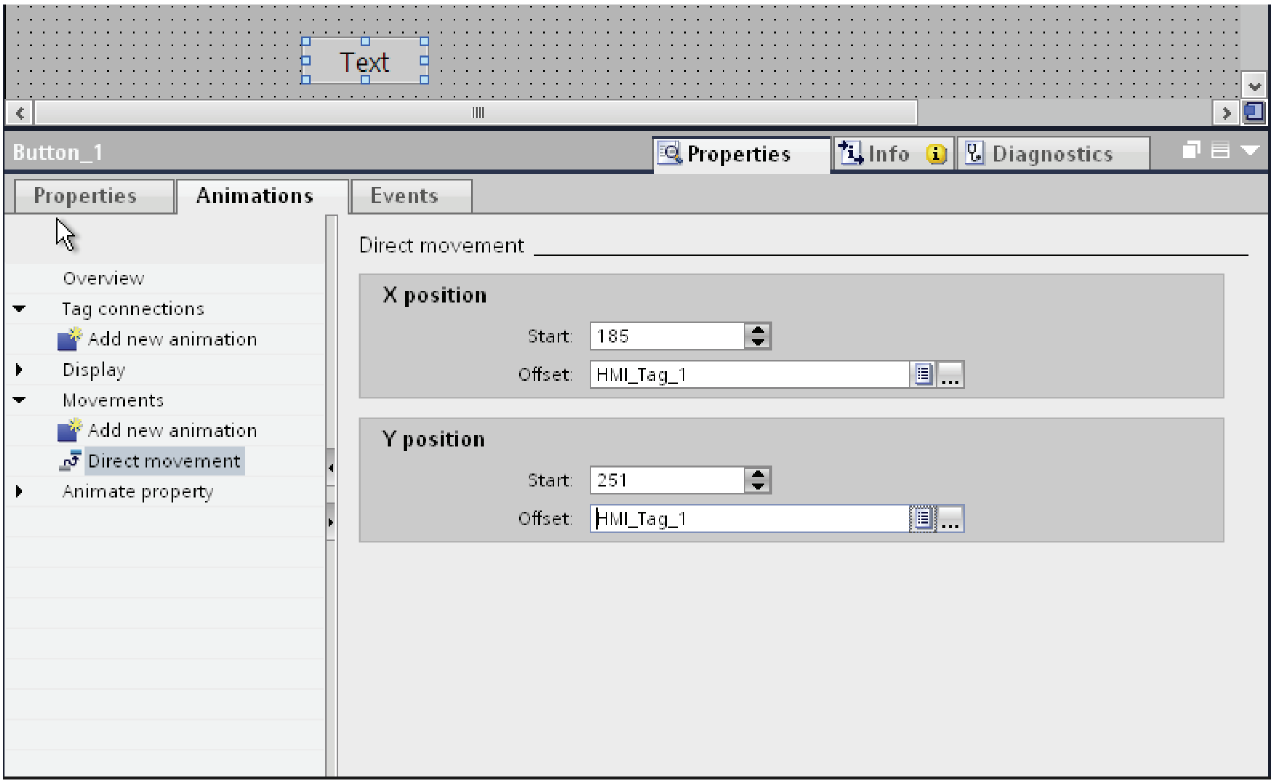
- Two tags are set up.
- You have opened a screen which contains at least one dynamic object.
- The Inspector window is open.
- The toolbox window is displayed.

Configuring "Direct movement"

1. Select the screen object you want to control dynamically.

   The object properties are displayed in the Inspector window.
2. In the Inspector window, select "Properties > Animations".
3. Select "Direct movement" and click the button.

   The parameters of the animation are displayed.
4. Select a tag for the X position with which the movement in x direction is controlled.
5. Select a tag for the Y position with which the movement in y direction is controlled.


Button_1

Direct movement

X position
Start: 185
Offset: HMI_Tag_1

Y position
Start: 251
Offset: HMI_Tag_1

Result

In Runtime, the object moves in response to every change in value of the tag that is used to control the movement.

Dynamizing the visibility of an object

Introduction

Dynamization of the "Visibility" property allows you to output an alarm to your screen, which is triggered when the tag value exceeds a critical range, for example. The alarm is cleared when the tag value returns to the non-critical range.

The "Simple recipe view" and "Simple alarm view" objects are always visible.

Requirement

- You have created a tag.
- You have opened a screen containing an object that you want to show or hide in Runtime.
- The Inspector window is open.

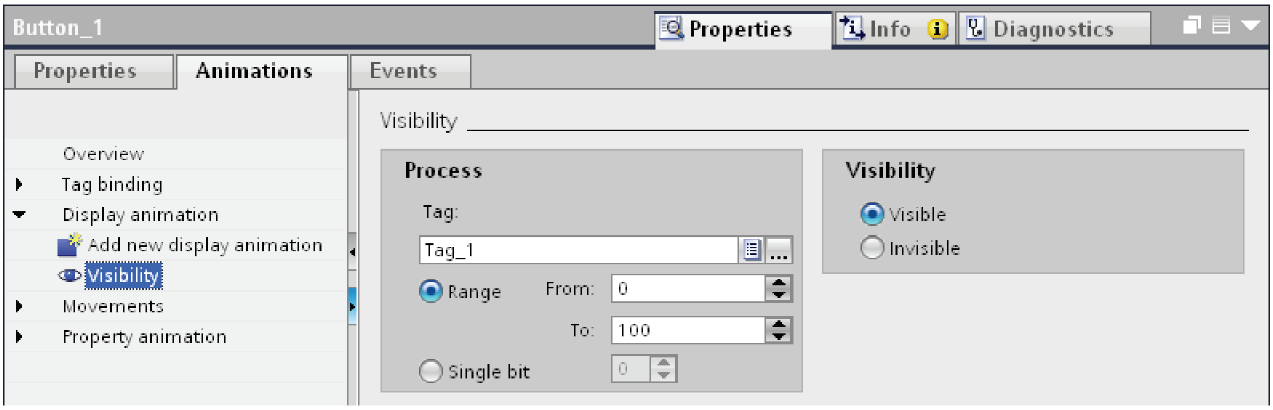
Procedure

1. Select the screen object you want to control dynamically.

   The object properties are displayed in the Inspector window.
2. In the Inspector window, select "Properties > Animations".

   The animations available for the selected object are displayed.
3. Select "Visibility" and click the button.

   The parameters of the animation are displayed.
4. Select a tag.
5. Activate "Single bit".
6. Select bit number "6" for example.
7. Activate "Visible".



Visibility

Process

Tag: Tag_1

● Range From: 0 To: 100

○ Single bit 0

Visibility

● Visible

○ Invisible

Result

The screen object is shown or hidden in Runtime depending on the tag value.

- If the tag value matches the configured bit number exactly, the screen object is shown.
- If the tag value matches the configured bit number exactly, the screen object is hidden.

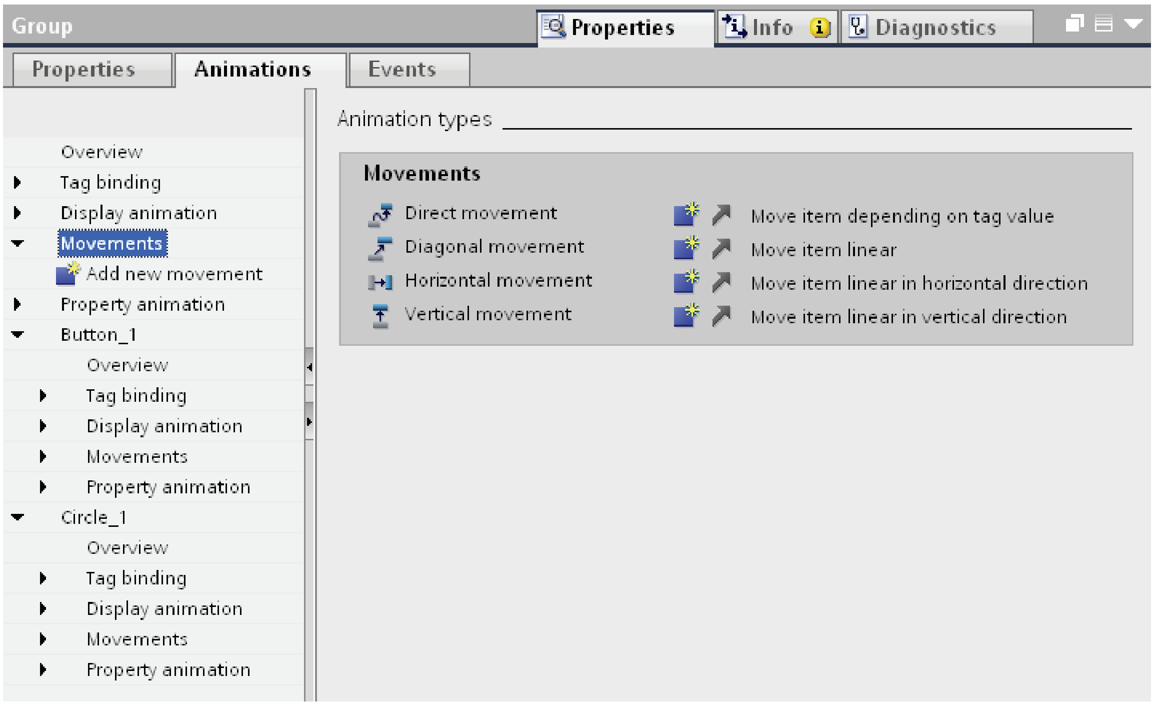
See also

Configuring a new animation (Page 2040)

Animations of object groups

Applying animations to object groups

The inspector window shows all objects of a group and their possible animations. The animation types which are supported by all objects in the group are also listed separately.



Direct movement	Move item depending on tag value
Diagonal movement	Move item linear
Horizontal movement	Move item linear in horizontal direction
Vertical movement	Move item linear in vertical direction

If you configure an animation for an object group, this animation will apply to all individual objects that support this animation.

Application example

The "horizontal movement" animation is configured for the object of an object group. The "direct movement" animation is configured for the whole object group. Only the animation of the object group i.e. "direct movement" is executed in Runtime. This also applies for object groups within object groups. Only the animation of the group on the top layer is listed.

Animations for object groups and for multiple selection

Changing animations for multiple objects

For a multiple selection, the animations that are configured for the reference object appear in the Inspector window. You can change the animations as usual. The changes will apply to all the objects in the multiple selection that support the configured animation. This means that even objects for which you have not yet configured an animation will have the reference object's animation.

Application example

Select a button, and a circle at the same time. The button is the reference object. The "Appearance" animation is already configured for the button, so it appears in the Inspector window of the multiple selection. When you activate "Properties > Animations >Appearance > Flashing" in the inspector window, the settings of the "Appearance" animation apply for the button and for the circle.

Configuring new animations for multiple objects

If you configure a new animation for the objects of a multiple selection, this animation will apply to all selected objects that support the configured animation. If the new animation replaces an existing animation, a security prompt appears.

Application example

You select a circle, and a rectangle. The "Diagonal movement" animation is already configured for the circle. You configure the "Horizontal movement" animation for the multiple selection. This animation applies to the rectangle since no animation of the Movement type is yet configured for it. For the circle, you are asked to confirm that you want to replace the existing "Diagonal movement" animation with the new "Horizontal movement" animation.

10.1.4.4 Dynamize with system functions

Basic on events

Introduction

Screen objects react to events. You configure a function list with system functions on the events of an object.

Events

Which events and system functions are available depends on the object used.

If the operator activates a screen object for example, the configured system function is executed.

Further information can be found in Working with function lists (Page 2287)

See also

Basics on dynamization (Page 2038)

Example: Configuring a button for language switching (Page 2050)

Configure system function on the "Click" event (Page 2049)

Configure system function on the "Click" event

Introduction

You configure a function list on an object event. The linked system function is executed when the event occurs in runtime.

Requirements

A screen is open.

A button has been created in the screen.

The inspector window is open.

Procedure

1. Select the button.
2. Click "Properties> Events" in the Inspector window.
3. Select the "Click" event.
4. Click "Add function" in the table.
5. Select the "ShowAlarmWindow" system function.

Result

The alarm window opens in the screen if the operator clicks the button in runtime.

See also

Basic on events (Page 2049)

Example: Configuring a button for language switching

Introduction

In this example, you configure a button that can be used to toggle between multiple runtime languages during runtime.

Requirements

- You have completed the "Configuring a button in multiple languages" example.
- The "Screen_1" screen is open.
- The button on the screen has been selected.

Procedure

1. In the Inspector window, select "Properties > Events > Press".
2. Click on "Add function" in the table.
3. Select the "SetLanguage" system function.

Result

You have assigned the button the function "SetLanguage". Pressing the button during runtime will switch the runtime language. The runtime languages are switched in the order specified by the number sequence in the "Languages and fonts" editor.

See also

Basic on events (Page 2049)

10.1.5 Working with function keys

10.1.5.1 Working with function keys

Introduction

The function key is a physical key on your HMI device and its functions can be configured. A function list can be configured for the events "Key pressed" and "Release key".

A function key can be assigned global or local functions.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Global function keys

Global function keys always trigger the same action, regardless of the displayed screen.

Global function keys are configured in the "Global Screen" editor. The global assignment applies for all the screens of the set HMI device.

Global function keys reduce programming considerably, because there is no need to assign these global keys to each individual screen.

Local function keys in screens

Local function keys in screens can trigger a different action in each screen. This assignment applies only to the screen in which you have defined the function key.

Using a local function key you can overwrite global function keys and the local function keys of a template.

Note

If a screen with local function keys is overlapped by an alarm view or an alarm window, then the function keys are still active in runtime. This may happen in particular on HMI devices with a small display.

Local function keys in templates

Local functions keys that are assigned in templates are valid for all the screens based on this template. They can trigger a different action in every screen. Function keys for templates are assigned in the template of the "Screens" editor. You overwrite the global assignment of a function key by a local assignment in the template.

Hotkey assignment

You can assign hotkeys, such as buttons, to control objects. The available hotkeys depend on the HMI device.

Note

The function key has a local or global action assigned to it. If the function key also has a hotkey assigned to it, then the hotkey function will be executed in Runtime.

Graphics

When a function key is placed directly next to the display, you can assign a graphic to it to make the function of the function key more clear.

Display of assignment

Table 10- 1 The following table shows which symbols display the assignment of the function keys:

Function key	Description
F1	Not assigned
F2	Global assignment
F6	Assigned locally in the template
F3	Local assignment
F8	Local assignment (local assignment of the template overwrites global assignment)
F4	Local assignment (local assignment overwrites global assignment)
F6	Local assignment (local assignment overwrites local assignment of the template)

Function key	Description
F8	Local assignment (local assignment overwrites local assignment of the template, which already overwrites global assignment)
F5	Assigning buttons with screen navigation

Note

Basic Panels

The "Screen Navigation" editor is not available for Basic Panels.

10.1.5.2 Assigning function keys globally

Introduction

You define the global assignment of a function key in the "Global Screen" editor. The global assignment applies to all screens of the set HMI device.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Requirement

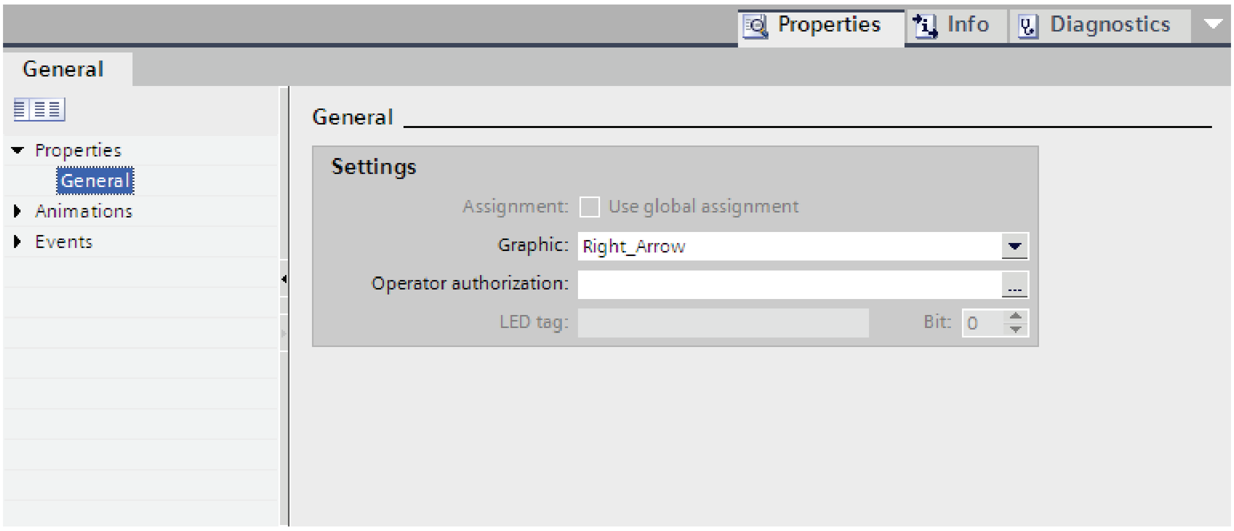
- You have opened the project.
- The Inspector window is open.

Procedure

Proceed as follows to assign a screen-based function to a function key:

1. To open the "Global Screen" editor, double-click "Global Screen" in the "Screen management" group of the Project window.
2. Select the desired function key.

   The properties of the function key are shown in the Inspector window.



Settings

Assignment: □ Use global assignment

Graphic: Right_Arrow

Operator authorization:

LED tag:

Bit: 0

3. Under "General", configure a graphic and an operator authorization for the function key.
4. Configure a function list for the required event under "Events".

Result

If a local assignment does not overwrite the global assignment, the assignment of the function key changes in all the screens of the set HMI device in accordance with your entry.

10.1.5.3 Local assignment of function keys

Introduction

Function keys are assigned globally and locally. A local assignment of the function keys is only valid for the screen or the template in which it was defined. The following local function keys are available:

- Local function keys of a screen

  Different functions are assigned to the function key for each screen. This assignment applies only to the screen in which you have defined the function key.
- Local function keys of a template

  You assign the function keys in a template. The assignment applies to all the screens that are based on this template and are not overwritten by a local assignment in a screen.

A local assignment overwrites the global assignment of a function key.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Using existing assignments

The option for using existing assignments is referred to as follows in the Inspector window:

- In a template: "Use global assignment"
- In a screen:
  - Screen based on a template: "Use local template"
  - Screen not based on a template: "Use global assignment"

The "Use local template" option includes the use of the local assignment in the template and the global assignment.

Requirement

- You have opened the screen or the template in which you want to assign a function key locally.
- The Inspector window is open.

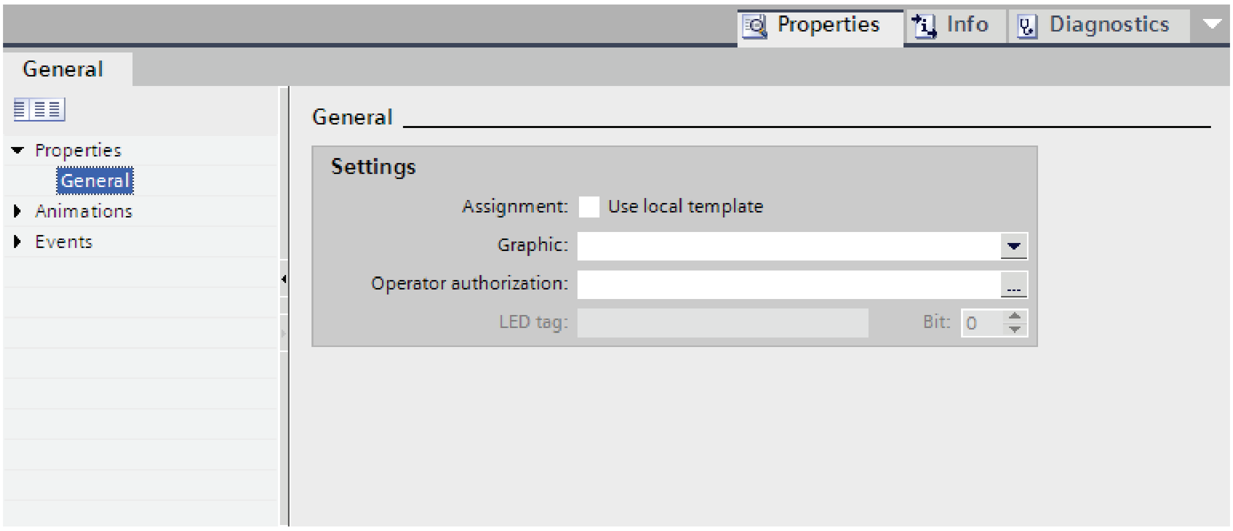
Procedure

Proceed as follows:

1. Select the desired function key in the screen or in the template.

   The properties of the function key are shown in the Inspector window.

2. Click "General" in the Inspector window.



Assignment: □ Use local template
Graphic:
Operator authorization:
LED tag:
Bit: 0

3. Disable the "Use local template" or "Use global assignment" option.
4. Under "General", configure a graphic and an operator authorization for the function key.
5. Configure a function list for the required event under "Events".

Result

The function key is assigned the configured function in the screen or in the template.

10.1.5.4 Assigning a function key to a function

Introduction

A function key can have two states:

- Pressed: Defined by the "Key pressed" event.
- Released: Defined by the "Release key" event.

Both of these events are configured in the Inspector window of the function key. You can assign any event a function list which contains system functions or scripts. Execution of this function list is event-driven in runtime.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Note

Basic Panels

Scripts are not available for Basic Panels.

Requirement

To assign the function key a global function:

- The "Global Screen" editor is open.

To assign the function key a local function:

- The screen in which you want to assign a function key is open.

If you want to assign a function key locally in a template:

- The template in which you want to assign a function key is open.
- The Inspector window is open.

Procedure

Proceed as follows:

1. Select the function key you want to define.

   The properties of the function key are shown in the Inspector window.
2. Configure the function list for the desired result in the Inspector window under "Properties" in the "General" group.

Result

The function list is executed in runtime when the operator presses or releases the function key.

10.1.5.5 Assigning operator authorization for a function key

Introduction

In WinCC you can assign an operator authorization for a function key in runtime. This allows you to restrict access to the function keys to specific persons or operator groups when you configure your project. Only authorized personnel can then change important parameters and settings in runtime.

You configure access protection in order to avoid operator errors and increase plant or machine security.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Requirement

- The user groups have been defined.

To protect a global function key:

- The "Global Screen" editor is open.

If you want to protect a local function key of a screen or of a template:

- The screen or the template which contains the function key is open.
- The Inspector window is open.

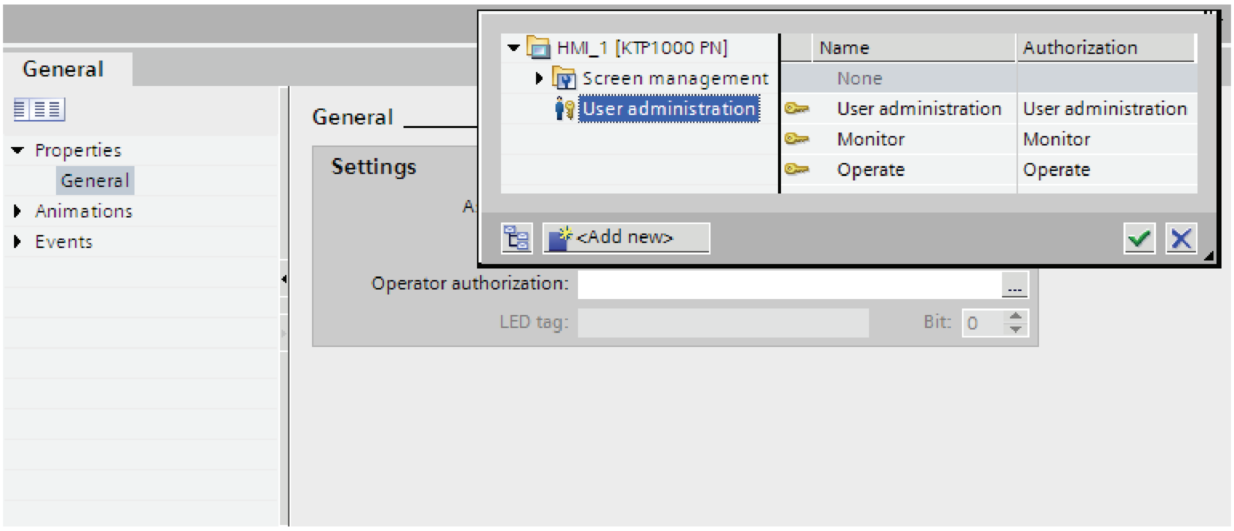
Procedure

Proceed as follows:

1. Select the relevant function key.

   The properties of the function key are shown in the Inspector window.

2. Click "General" in the Inspector window.



Name	Authorization
None	
User administration	User administration
Monitor	Monitor
Operate	Operate

Operator authorization:

LED tag: Bit: 0

3. In the "Authorization" list, select the user group you want to allow runtime access to the function key.

Result

The operator authorization is configured.

10.1.5.6 Assigning a function key to a graphic

Introduction

In order to make the function of a key more clear, you can insert a graphic in the screen alongside the function key. Graphics can only be assigned to function keys that border the screen margin of the HMI device.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Requirement

To assign a graphic to a global function key:

- The "Global Screen" editor is open.

If you want to assign a graphic to a local function key in a screen or template:

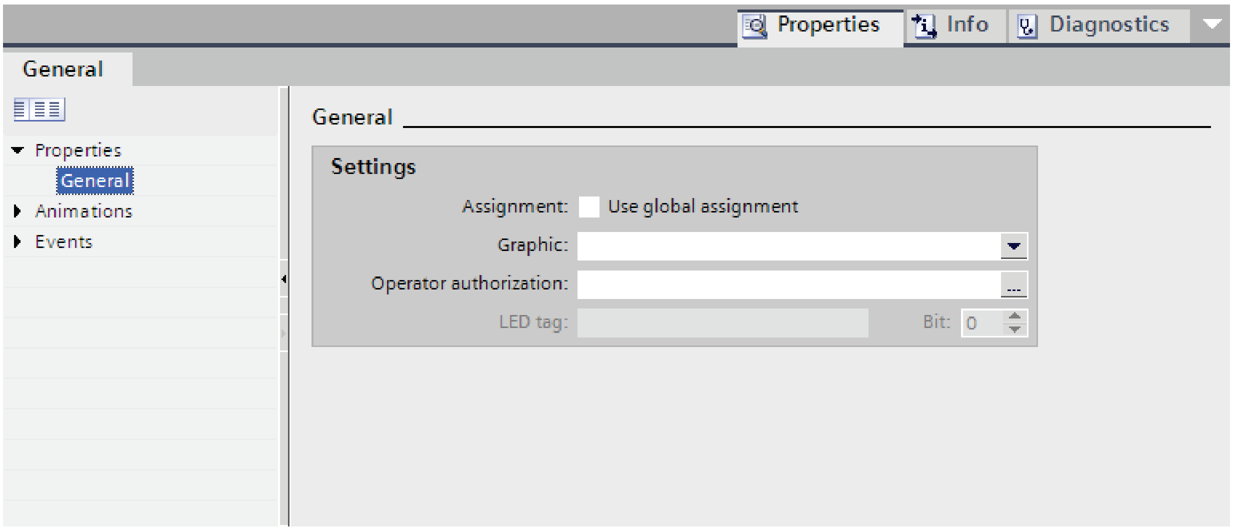
- The screen or the template that contains the corresponding function key is open.
- The Inspector window is open.
- You have created the graphic for the function key.

Procedure

Proceed as follows:

1. Select the relevant function key.

   The properties of the function key are shown in the Inspector window.
2. Click "General" in the Inspector window.



Settings

Assignment: □ Use global assignment

Graphic:

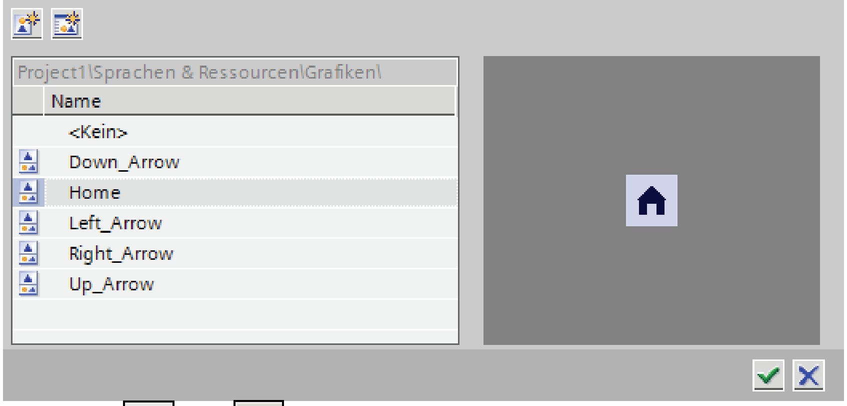
Operator authorization:

LED tag:

Bit: 0

3. Click in the "Graphic" area of the list.

   The graphic browser of your WinCC project appears. The pane on the left side shows the external graphics which are stored in the browser. The pane on the right side shows you a preview of the graphic you have selected in the browser.

   Using the ▦ and ▦ icons, you can display the collection either in form of thumbnails or as a list.

   In order to open and edit OLE objects with the associated graphics program, double-click on the object.

4. In the browser click the desired graphic or store the relevant graphic in the graphic browser.

   The graphic preview is shown in the right pane.

5. Click "Select" to add the graphic to the screen.

   Click "Clear" to remove the graphic from the screen.

Result

The graphic is displayed next to the function key.

10.1.5.7 Example: Using function keys for screen navigation

Task

In this example you create a local function key in a screen. When the operator presses this function key, a screen change to a predefined screen is triggered, for example "Boiler 2".

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Requirement

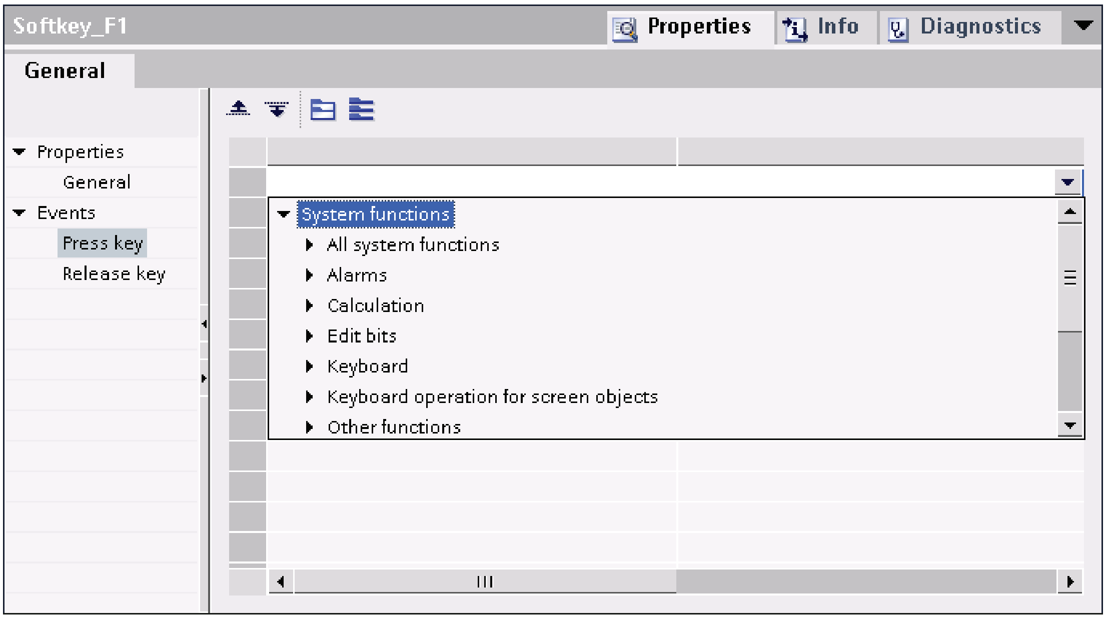
- The screen in which you want to assign the function key is open.
- You have created the "Boiler 2" screen.
- The Inspector window is open.

Procedure

Proceed as follows to use the "ActivateScreen" function:

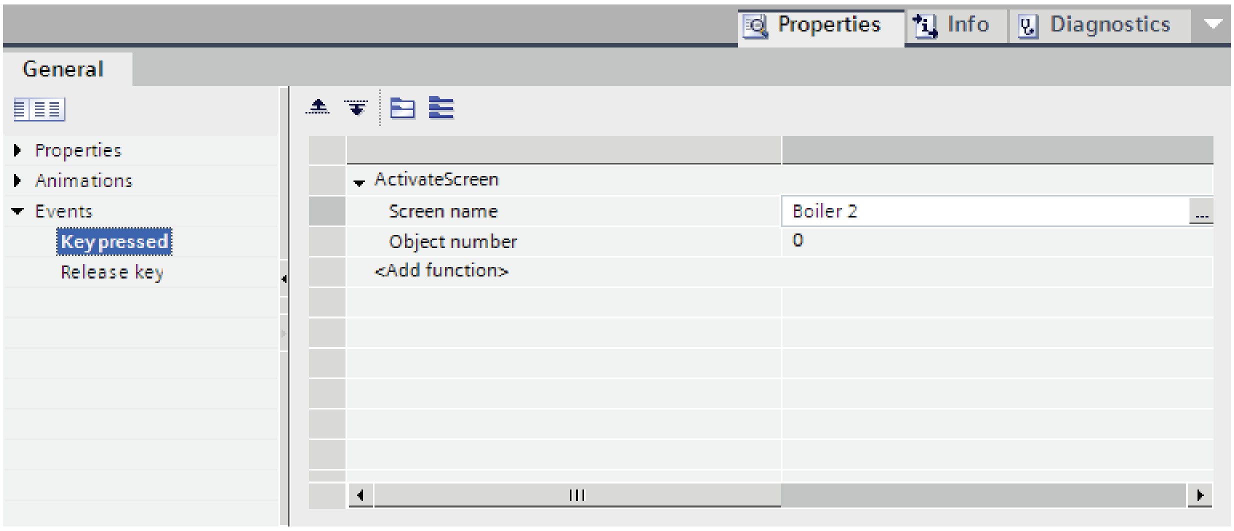
1. Select the desired function key.

   The properties of the function key are shown in the Inspector window.
2. Click "General."
3. To overwrite a global assignment, disable the "Use local template" option.
4. Click "Key pressed" under "Events".



5. Select the "ActivateScreen" system function from the list.

   The "ActivateScreen" function appears in the "Function list" dialog box, including the "Screen name" and "Object number" parameters.



ActivateScreen	
Screen name	Boiler 2
Object number	0
<Add function>	

6. Select the "Boiler 2" screen from the "Screen name" list.

Result

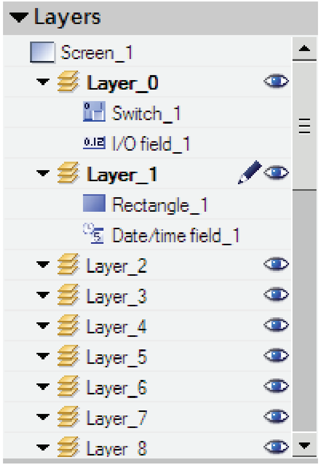
The operator changes to the "Boiler 2" screen in runtime by pressing the selected function key.

10.1.6 Working with layers

10.1.6.1 Basics on working with layers

Layers

Use layers in order to achieve differentiated editing of the objects in a screen. A screen consists of 32 layers that you can give any names. If you assign objects to the layers, you thereby define the screen depth. Objects of layer 0 are located at the screen background, while objects of layer 31 are located in the foreground.



The objects of a single layer are also arranged hierarchically. When you create a screen, the object inserted first is located at the rear within the layer. Each further object is placed one position towards the front. You can shift objects forwards and backwards within a layer.

Principle of the layer technique

Always one layer of the 32 layers is active. New objects you add to the screen are always assigned to the active layer. The number of the active level is displayed in the inspector window of the screen and in the "Layout > Layers" task card.

When you open a screen, all 32 layers of the screen are displayed. You can hide all the layers except for the active layer in the inspector window of the screen and in the "Layout > Layers" task card. You then explicitly edit objects of the active layer.

In the tree view of the "Layers" palette in the "Layout" task card, you administer layers and objects with drag-and-drop and the context menu.

Application examples

Use layers, for example, in the following cases:

- To hide the labeling of objects when editing,
- To hide objects, e.g. alarm windows, while configuring other objects

10.1.6.2 Moving objects between layers

Introduction

By default, a new object is inserted on the active layer. You can, however, assign an object to another layer at a later time.

Requirement

- A screen with an object is open.
- The Inspector window is open.

Procedure

1. Select the object in the screen.

   The object properties are displayed in the Inspector window.
2. Enter the layer to which you want to move the object in "Properties > Properties > Miscellaneous > Layer" in the Inspector window.

Alternatively, select the object from the "Layout" task card and drag it to the required layer.

Changing the order of objects

1. Select the object in the screen.

   The object properties are displayed in the Inspector window.
2. To move the object to the front or back, select the "Order" > "Move backward" or "Move forward" command from the shortcut menu.

   Alternatively, use the ↓ or ↑ button in the toolbar.

Result

The object is assigned to the selected layer, and positioned at the top of the layer.

10.1.6.3 Setting the active layer

Introduction

The screen objects are always assigned to one of the 32 layers. There is always an active layer in the screen. New objects you add to the screen are always assigned to the active layer.

The number of the active layer is indicated in the "Layer" toolbar. The active layer is indicated by the ✏ icon in the "Layout > Layers" task card.

Layer 0 is the active layer when you start programming. You can activate a different layer during configuration, if necessary.

Requirement

- You have opened a screen which contains at least one object.
- The Inspector window of the active screen is open.

Procedure

1. Click "Properties > Properties > Layers" in the Inspector window of the current screen.
2. Enter the layer number in "Settings > Active layer".

Alternative procedure

1. Select "Layout > Layers" in the "Layout" task card.
2. Select the "Set to active" command from the shortcut menu of a layer.

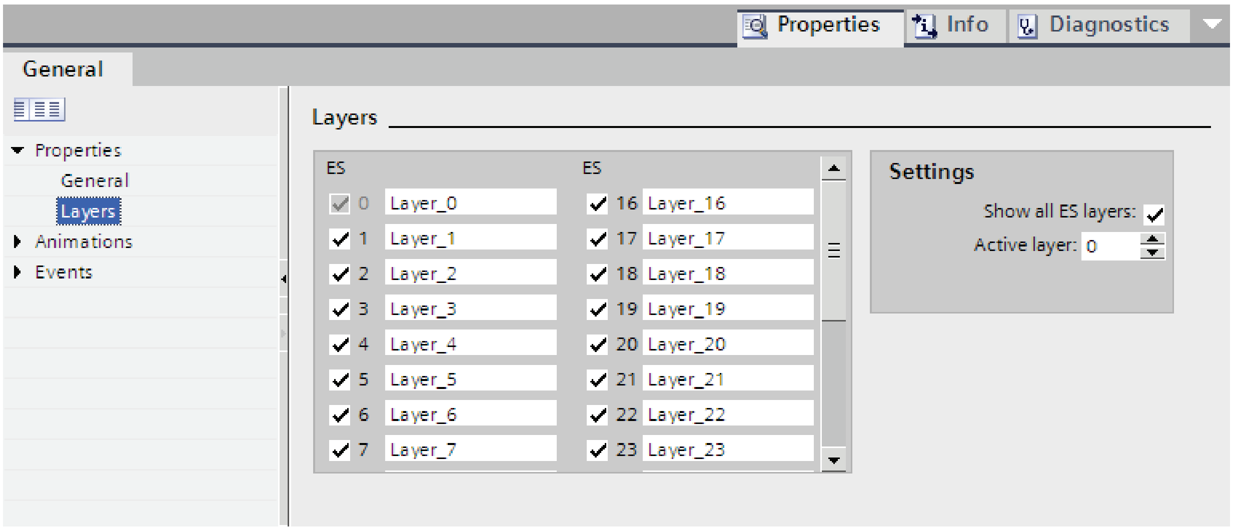
Result

The layer with the specified number is now active.

10.1.6.4 Show and hide layers

Introduction

You can show or hide the layers of a screen as required. You specify the layers that are shown in the Engineering System. When you open a screen, all the layers are always shown.



Layers

ES			ES		
✓	0	Layer_0	✓	16	Layer_16
✓	1	Layer_1	✓	17	Layer_17
✓	2	Layer_2	✓	18	Layer_18
✓	3	Layer_3	✓	19	Layer_19
✓	4	Layer_4	✓	20	Layer_20
✓	5	Layer_5	✓	21	Layer_21
✓	6	Layer_6	✓	22	Layer_22
✓	7	Layer_7	✓	23	Layer_23

Settings

Show all ES layers: ☑

Active layer: 0

Requirement

- The screen is opened.
- The "Layout" task card is open.

Procedure

1. Select the layer that you want to hide or show in the "Layout > Layers" task card.
2. Click one of the icons next to the corresponding layer:
   - 👁 A shown layer is hidden
   - 👁 A hidden layer is shown

Note

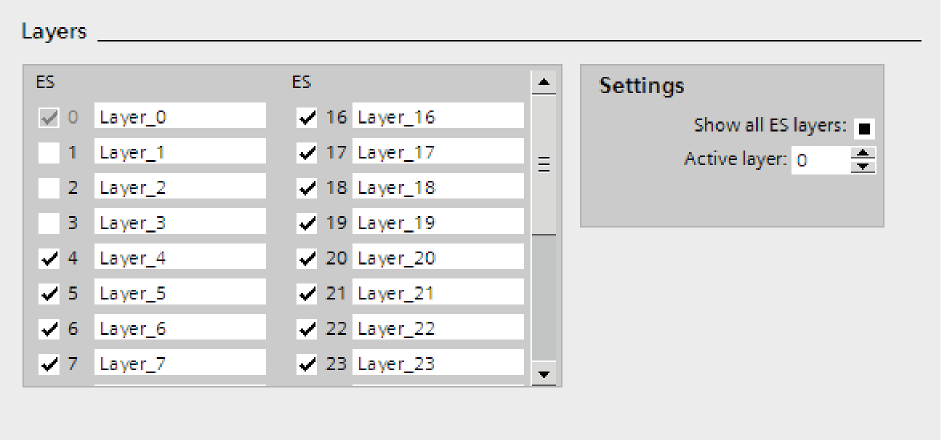
The active layer cannot be hidden.

Alternative procedure

1. Click in an area of the screen that does not contain an object.

   The screen properties are shown in the Inspector window.

2. In the Inspector window, select "Properties > Properties > Layers":



Layers

ES			ES		
☑	0	Layer_0	☑	16	Layer_16
□	1	Layer_1	☑	17	Layer_17
□	2	Layer_2	☑	18	Layer_18
□	3	Layer_3	☑	19	Layer_19
☑	4	Layer_4	☑	20	Layer_20
☑	5	Layer_5	☑	21	Layer_21
☑	6	Layer_6	☑	22	Layer_22
☑	7	Layer_7	☑	23	Layer_23

Settings

Show all ES layers: ■

Active layer: 0

3. In the list, disable the levels you wish to hide.

   If you activate "All ES layers" for a layer, the objects in this layer will be shown in the Engineering System.

Result

The layers are shown according to your settings.

10.1.6.5 Renaming layers

Introduction

When you create a screen, the 32 layers are numbered consecutively by default. To improve clarity, you can rename the layers to suit your requirements.

Requirement

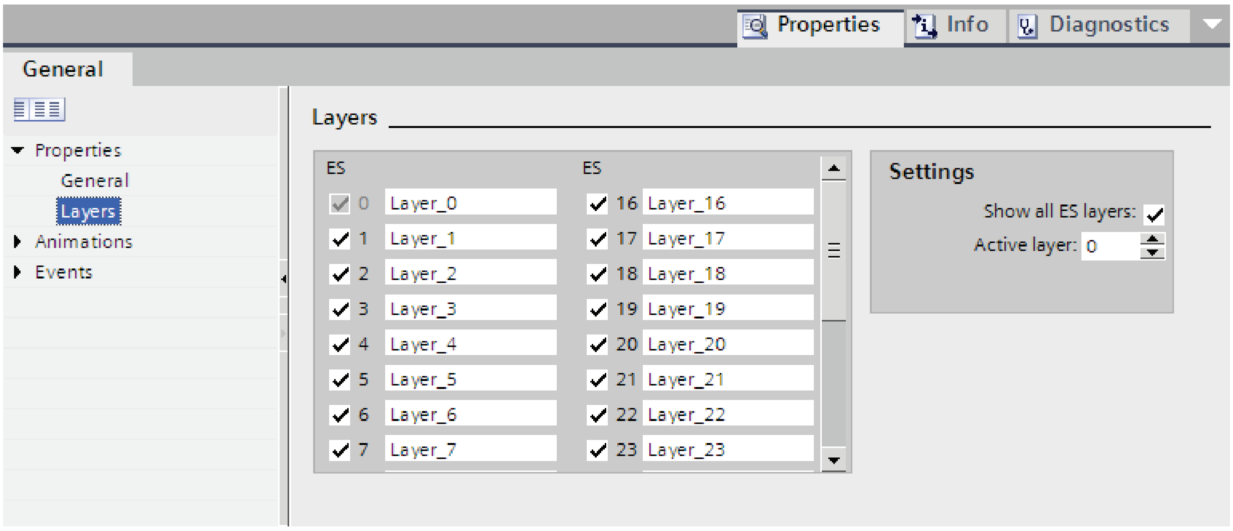
- The screen is opened.

Procedure

1. Click in an area of the screen that does not contain an object.

   The screen properties are shown in the Inspector window.

2. In the Inspector window, select "Properties > Properties > Layers".



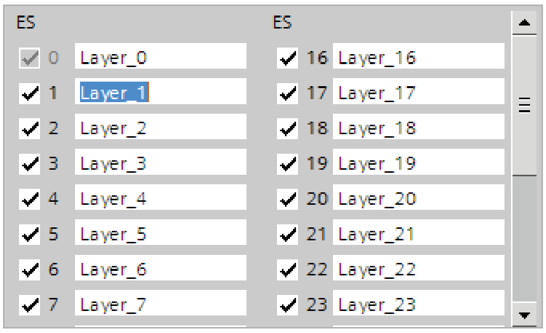
ES			ES		
✓	0	Layer_0	✓	16	Layer_16
✓	1	Layer_1	✓	17	Layer_17
✓	2	Layer_2	✓	18	Layer_18
✓	3	Layer_3	✓	19	Layer_19
✓	4	Layer_4	✓	20	Layer_20
✓	5	Layer_5	✓	21	Layer_21
✓	6	Layer_6	✓	22	Layer_22
✓	7	Layer_7	✓	23	Layer_23

Settings

Show all ES layers: ✓

Active layer: 0

3. Enter the new layer name.



ES			ES		
✓	0	Layer_0	✓	16	Layer_16
✓	1	Layer_1	✓	17	Layer_17
✓	2	Layer_2	✓	18	Layer_18
✓	3	Layer_3	✓	19	Layer_19
✓	4	Layer_4	✓	20	Layer_20
✓	5	Layer_5	✓	21	Layer_21
✓	6	Layer_6	✓	22	Layer_22
✓	7	Layer_7	✓	23	Layer_23

Result

The layer is displayed with the new name.

10.1.7 Working with libraries

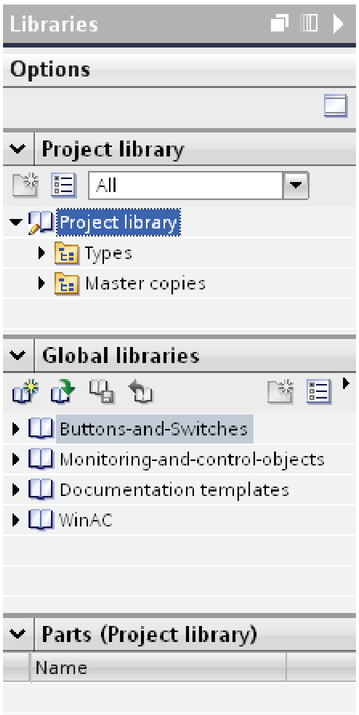
10.1.7.1 Basics on libraries

Introduction

Store all objects you need frequently in the libraries. An object that is stored in the library only has to be configured once. It can then be used repeatedly as often as required. Library objects extend the number of available screen objects and increase the effectiveness during configuration through the multiple usage of ready-to-use objects.

Libraries are managed in the "Libraries" task card. The following libraries are available:

- Project library
- Global libraries



Note

There is a symbol library in the "Tools" task card in the "Graphics" palette.

Project library

There is one library for each project. Objects of the project library are stored alongside with the project data and are available only for the project in which the library was created. If the project is moved to another PC, any project library created in it is also moved.

To use the library object of the project library in other objects, move or copy the object into a global library.

Global libraries

In addition to the objects from the project library, you can also incorporate objects from global libraries in your projects. A global library is saved independently of the project data in its own file with the extension *.al11.

A project can access several global libraries. A global library may be used concurrently in several projects.

When a library object is changed by a project, this library will be changed in all projects in which these libraries are open.

Library objects

A library can contain all WinCC objects. Examples:

- Complete HMI device
- Screens
- Display and control objects including tags and functions
- Graphics
- Tags
- Alarms
- Text and graphics lists
- Faceplates
- Structures

See also

Copy templates and types (Page 2072)

10.1.7.2 Copy templates and types

Introduction

Both the "Project library" and the "Global library" contain the two folders "Copy templates" and "Types". You can create or use the library objects either as a copy template or a type.

Copy templates

Use copy templates to create independent copies of the library object.

Types

Create instances of objects of the "Types" folder and use these in your project. The instances are bound to their respective type. Changes to an instance also change all other instances. Types are marked by a green triangle in the "Libraries" task card.

Administration of the library objects

You can only copy and move library objects within the same library. You can only copy copy templates to the "Copy templates" folder or any sub-folder of "Copy templates". You can also only insert types in the "Types" folder or any sub-folder of "Types".

See also

Basics on libraries (Page 2070)

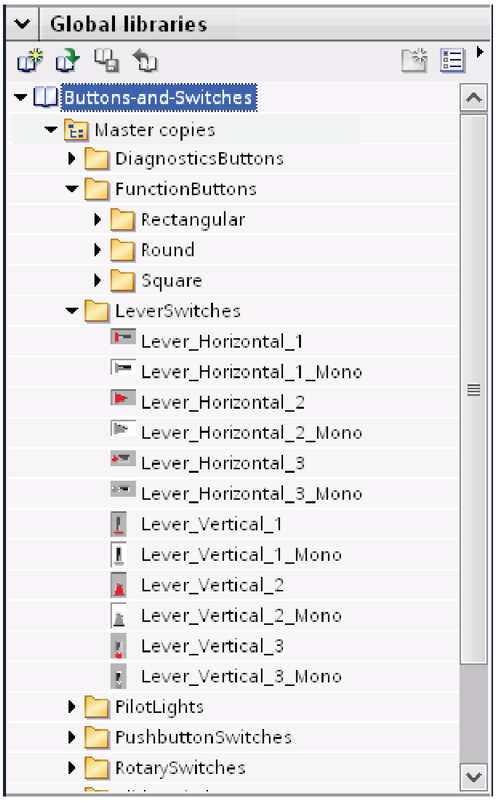
10.1.7.3 Libraries in WinCC

Introduction

The WinCC software package includes extensive libraries. Sorted by topic into folders, they contain preassembled graphic objects which you can use in screens for operation and monitoring of your plant.

Global library "Buttons and Switches"

The libraries "Buttons and Switches" offer a wide selection of buttons and switches.



The folders divide switches and buttons into categories. The "DiagnosticsButtons" folder contains the object "System diagnostics indicator", for example. You use the "System diagnostics indicator" object for system diagnostics in your plant.

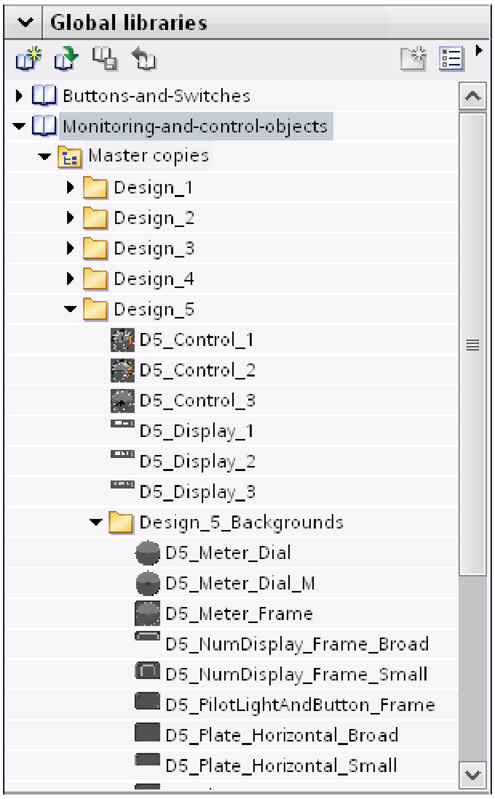
Note

You can only use the objects in the "DiagnosticsButtons" folder on Comfort Panels.

You cannot use objects which have "Switch" in the object name or in the associated folder name in Runtime Professional.

Global library "Monitoring and Control objects"

The "Monitoring and Control objects" library offers more complex display and operating objects in several designs and corresponding control lights, buttons and switches.



Global libraries

- Buttons-and-Switches
- Monitoring-and-control-objects
  - Master copies
    - Design_1
    - Design_2
    - Design_3
    - Design_4
    - Design_5
      - D5_Control_1
      - D5_Control_2
      - D5_Control_3
      - D5_Display_1
      - D5_Display_2
      - D5_Display_3
      - Design_5_Backgrounds
        - D5_Meter_Dial
        - D5_Meter_Dial_M
        - D5_Meter_Frame
        - D5_NumDisplay_Frame_Broad
        - D5_NumDisplay_Frame_Small
        - D5_PilotLightAndButton_Frame
        - D5_Plate_Horizontal_Broad
        - D5_Plate_Horizontal_Small

In addition, graphics views for the designs are stored in the "Design_Backgrounds" folder; they can be used as object backgrounds for the custom extension of the scope of the library.

Note

You cannot use objects which have "Switch" in the object name in Runtime Professional. The same applies for the object "D5_Display_3" with the date/time field it contains.

10.1.7.4 Displaying library objects

Introduction

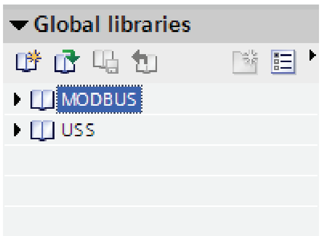
The libraries are displayed as file folders in the corresponding palette. The elements contained in the library are displayed in the file folder and in the "Elements" palette.

Requirement

- At least one library object has been created in a library.
- The "Libraries" task card is opened.

Procedure

1. Select the library in the corresponding palette whose library objects you want to display.

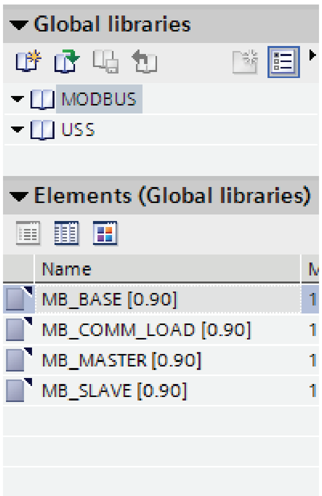


Global libraries

- MODBUS
- USS

2. Click ☰.

   The contained library objects are displayed in the "Elements" palette.



Global libraries

- MODBUS
- USS

Elements (Global libraries)

Name	
MB_BASE [0.90]	1
MB_COMM_LOAD [0.90]	1
MB_MASTER [0.90]	1
MB_SLAVE [0.90]	1

3. Click one of the following icons:

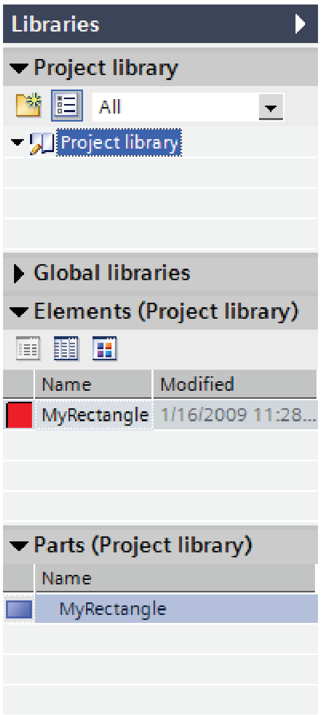
Icon	Description
▤	Element view in detailed mode
▥	Element view in list mode
▦	Element view in overview mode with icons

When several objects are assigned to the library with a multiple selection, only one of the objects is shown in the "Elements" palette. The individual components of this element are displayed in the "Parts" palette.

Show parts of the library objects

1. Select the library in the corresponding palette from which you want to view the components of an element.
2. Click ▤.
3. The contained library objects are displayed in the "Elements" palette.
4. Select the element.

   The "Parts" palette shows the objects of which the element consists.



	Name	Modified
🟥	MyRectangle	1/16/2009 11:28...

	Name
🟦	MyRectangle

Result

The library objects are displayed in accordance with the configuration. The components of the faceplates are displayed.

10.1.7.5 Managing library objects

Introduction

You can always copy or move library objects within the categories of a library. Delete the library objects you do not require.

Note

Copy templates and types

You can only copy and move library objects within the same library. You can only copy copy templates to the "Copy templates" folder or any sub-folder of "Copy templates". You can also only insert types in the "Types" folder or any sub-folder of "Types".

Requirement

- You have opened a library which contains several categories and at least one object.
- The library object is shown.

Moving a library object

1. Select the library object.
2. Drag the object to the desired folder with drag&drop.

Copying a library object

1. Select the library object.
2. Select "Copy" from the shortcut menu.
3. Select the folder in which you want to insert the library object.
4. Select "Paste" from the shortcut menu.

Deleting a library object

1. Select the library object.
2. Select "Delete" from the shortcut menu.

Renaming a library object

Proceed as follows to rename a library object

1. Click the object that you wish to rename with the right mouse button.
2. Select the "Rename" command from the shortcut menu.
3. Enter the new name.

10.1.7.6 Storing an object in a library

Introduction

You can store all of WinCC objects, such as screens, tags, graphic objects or alarms in your libraries. You can use drag-and-drop to move the corresponding object from the work area, project window or detail view to the library. In a library you have divided into categories, you can directly add objects to a specific category.

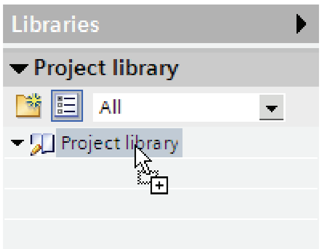
Requirement

- The "Screens" editor is open.
- A screen object has been created in the work area of the screen.
- The created libraries are displayed.

Procedure

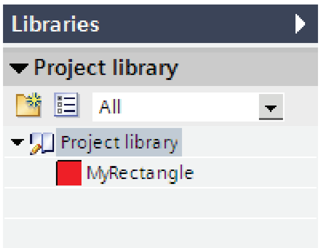
1. Select the object in the work area of the "Screens" editor.
2. Drag-and-drop the object from the work area to the desired library.

   The mouse pointer is transformed into a crosshair with an appended object icon.



Result

The object is saved to the library for further use in multiple instances of your configuration.



10.1.7.7 Inserting a library object

Introduction

The system always assigns the inserted library object a name which consists of the name of the object type and of a consecutive number. If the inserted object already exists, you can use a dialog window to specify whether or not the existing object should be replaced or inserted under a new name. Enter a new name if you do not want to replace the existing object.

You cannot insert library objects that are not supported by the HMI device.

Note

If you insert a screen with interconnected template from the library, the template will also be inserted. Any existing matching template is not used.

Requirement

- The "Libraries" task card is opened.
- The editor in which you want to insert the library object is open.

Procedure

1. Select the library object that you want to insert in the library.
2. Drag-and-drop the library object to the position in the work area where you want to insert the object.

   The library object is inserted.
3. Select the library object in the screen and adapt it.

Result

If the object was contained in the "Copy templates" folder, you have inserted an independent copy of the library object in the editor.

If the object was contained in the "Types" folder, you have inserted an instance of the library object in the editor.

10.1.7.8 Creating a global library

Introduction

In the libraries you store the configured objects that you want to use several times in your configuration. To use objects in several projects, create a global library.

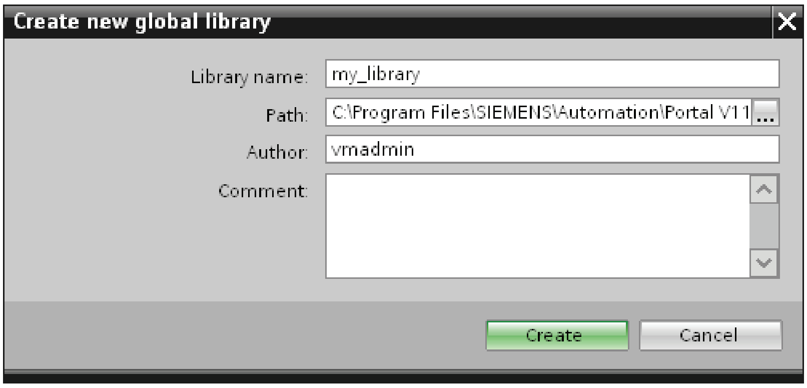
Requirement

- You have opened the project.
- The "Libraries" task card is opened.

Procedure

1. Click the icon in the "Libraries > Global libraries" task card.

   The "Create new global library" dialog opens.

   Create new global library

   Library name: my_library

   Path: C:\Program Files\SIEMENS\Automation\Portal V11

   Author: vmadmin

   Comment:

2. Enter a name.
3. Select the path where the new library is to be stored.
4. Click "Create".

Result

The new library is shown in the "Global libraries" palette. The global library contains the "Types" and "Copy templates" folders. You can save objects to the library.

10.1.7.9 Saving a global library

Introduction

A global library is stored in a separate file on your hard disk drive. The file contains the objects of the global library including the referenced objects. E.g. the reference of a tag which was configured on an I/O field is also saved in the library.

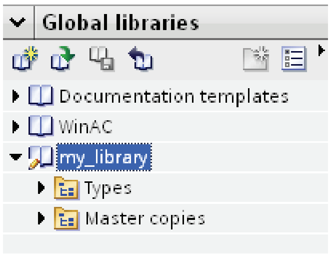
WinCC prompts you to save the global libraries when you close WinCC or your project without saving. You also can store the global library during configuration, without storing the entire project.

Requirement

- You have opened a project which contains at least one library.
- The "Libraries" task card is opened.
- A library has been changed.

Procedure

1. Select the global library that you want to save.



2. Click the 🗐 icon in the "Libraries" task card in the "Global libraries" palette.

You can alternatively select the "Save Library" command in the shortcut menu.

If you want to store the global library in a different folder, select "Save as" in the shortcut menu. Select the path in which you want to store the new library and enter a file name.

Result

The global libraries are saved under their current file name or a file name you have specified.

10.1.7.10 Opening a global library

Introduction

In WinCC, the global libraries are stored in separate files. You can use a global library in every project.

Requirement

- You have saved a global library.
- A project is open.
- The "Libraries" task card is opened.

Procedure

1. Click the icon in the "Global libraries" palette.

   The "Open global library" dialog box is displayed.
2. Select the path in which the library is stored.
3. Click "Open."

Note

If you want to access a global library from several projects, the global library will have to be read-only when you open it. As soon as a global library is read-only, access from other projects is blocked.

Result

WinCC displays the opened global library in the "Global libraries" palette.

10.1.8 Display and operating objects

10.1.8.1 Device-Specific Nature of the Objects

Objects for Basic Panels

Availability of display and operating elements for Basic Panels

Only the objects which can be used for the device you are configuring will be shown in the object window. The following table shows the availability of indicator and control objects for the Basic Panels.

Overview

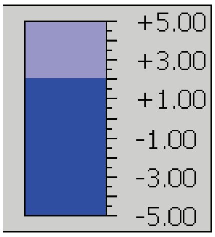
	KTP300 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	KTP1500 Basic
Bar	Yes	Yes	Yes	Yes	Yes
User view	Yes	Yes	Yes	Yes	Yes
Date/time field	Yes	Yes	Yes	Yes	Yes
I/O field	Yes	Yes	Yes	Yes	Yes
Ellipse	Yes	Yes	Yes	Yes	Yes
Graphic view	Yes	Yes	Yes	Yes	Yes
Graphic I/O field	Yes	Yes	Yes	Yes	Yes
Help indicator	Yes	No	No	No	No
Circle	Yes	Yes	Yes	Yes	Yes
Trend view	Yes	Yes	Yes	Yes	Yes
Line	Yes	Yes	Yes	Yes	Yes
Alarm view Alarm window	Yes	Yes	Yes	Yes	Yes
Alarm indicator	Yes	Yes	Yes	Yes	Yes
Rectangle	Yes	Yes	Yes	Yes	Yes
Recipe view	Yes	Yes	Yes	Yes	Yes
Button	Yes	Yes	Yes	Yes	Yes
Switch	Yes	Yes	Yes	Yes	Yes
Symbolic I/O field	Yes	Yes	Yes	Yes	Yes
Text field	Yes	Yes	Yes	Yes	Yes

10.1.8.2 Objects

Bar

Application

The tags are displayed graphically using the "Bar" object. The bar graph can be labeled with a scale of values.



Layout

In the Inspector window, you customize the settings for the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Color transition: Specifies the change in color display when limit values are exceeded.
- Displaying the limit lines / limit markers: Shows the configured limit as a line or marker.
- Define bar segments: Defines the gradations on the bar scale.
- Define scale gradation: Defines the subdivisions, scale markings and intervals of a bar scale.

Color transition

You define how the color change is represented in "Properties > Properties > Appearance" in the Inspector window.

Color transition	Description
"Segmented"	If a particular limit was reached, the bar changes color segment by segment. With segment by segment representation, you visualize, for example, which limits are exceeded by the displayed value.
"Entire bar"	If a particular limit was reached, the entire bar changes color.

Displaying limit lines and limit markers

You display the configured limit in the bar as a line or marking in Runtime using the "Lines" and "Markings" property:

1. In the Inspector window, select "Properties > Properties > Appearance":
2. Activate "Lines" and "Markings".

Define bar segments

Use the "Subdivisions" property to define the number of segments into which the bar is divided by the main gradations on the scale.

Use the "Interval" property to divide the distance between the main gradations. The value appears as the difference in value between two adjacent main gradations:

1. In the Inspector window, select "Properties > Properties > Scales":
2. Activate "Show scale."
3. Select the corresponding value for "Settings > Subdivisions".
4. Select the corresponding value for "Settings > Marks label".
5. Select the corresponding value for "Large interval > Interval".

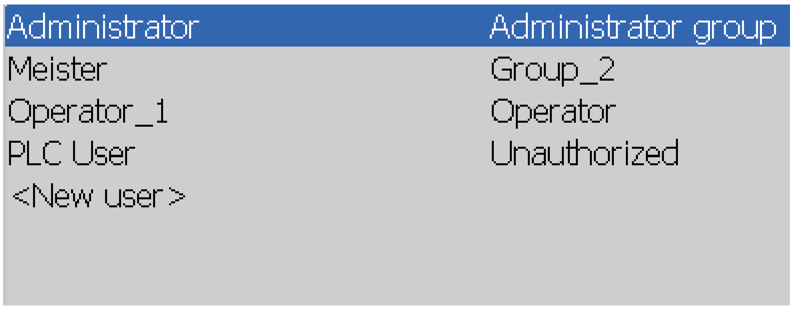
See also

Device-Specific Nature of the Objects (Page 2083)

User view

Application

The "User view" object is used to set up and administer users and authorizations.



Administrator	Administrator group
Meister	Group_2
Operator_1	Operator
PLC User	Unauthorized
<New user>	

Note

Do not use the simple user view in a group.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Number of lines: Specifies the maximum number of visible entries.

Number of lines

The number of lines in the user view displayed in Runtime is specified in the Inspector window. The setting for the number of lines is only effective if the property "Fit object to contents" is active.

1. In the Inspector window, select "Properties > Properties > View".
2. Enter an integer value under "Number of lines".
3. In the Inspector window, activate "Properties > Properties > Layout".
4. Activate "Fit object to contents."

See also

Device-Specific Nature of the Objects (Page 2083)

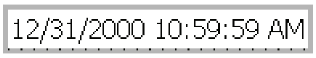
Simple user view (Page 2823)

Configuring a user view (Page 2267)

Date/time field

Application

The "Date/time field" object shows the system time and the system date. The appearance of the "Date/time field" depends on the language set on the HMI device.



12/31/2000 10:59:59 AM

Layout

In the Inspector window, you customize the position, style, colors and font types of the object. You can adapt the following properties in particular:

- Display system time: Specifies that the system time is displayed.
- Include tag: Specifies that the time of the connected tag is displayed.
- Long date/time format: This setting defines the format displayed for the data and time.

Display system time

The time displayed in the "Date/time field" on the HMI device is specified in the inspector window.

1. Click the "General" group in the inspector window.
2. Activate the "System time" option in the "Format" area.

Using tags

The time of the interconnected tag is displayed in the date/time field.

1. Click the "General" group in the inspector window.
2. In the "Format" area, select a tag with the "DateTime" data type, e.g. an internal tag. Information about data types which are available for connection to other PLCs can be found in the documentation about the respective communication drivers.

Long date/time format

The layout of the date and time is specified in the "Format" area under "General" in the inspector window.

Option	Description
"Enabled"	Date and time are displayed in full, e.g. "Sunday, December 31, 2000 10:59:59 AM"
"Disabled"	Date and time are displayed in short form, e.g. "12/31/2000 10:59:59 AM"

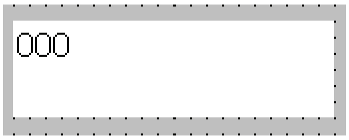
See also

Device-Specific Nature of the Objects (Page 2083)

I/O field

Application

The "I/O field" object is used to enter and display process values.



Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Mode: Specifies the response of the object in Runtime.
- Display format: Specifies the display format in the I/O field for input and output of values.
- Hidden input: Specifies whether the input value is displayed normally or encrypted during input.

Note

Reports

In reports, I/O fields only output data. "Output" mode is preset. Properties for configuring input are not available, e.g. "hidden input".

Mode

The response of the I/O field is specified in the Inspector window in "Properties > Properties > General > Type".

Mode	Description
"Input"	Values can only be input into the I/O field in runtime.
"Input/output"	Values can be input and output in the I/O field in runtime.
"Output"	The I/O field is used for the output of values only.

Layout

The "display format" for the input and output of values is specified in "Properties > Properties > General > Format" in the Inspector window.

Layout	
"Binary"	Input and output of values in binary form
"Date"	Input and output of date information. The format depends on the language setting on the HMI device.
"Date/time"	Input and output of date and time information. The format depends on the language setting on the HMI device.
"Decimal"	Input and output of values in decimal form
"Hexadecimal"	Input and output of values in hexadecimal form
"Time"	Input and output of times. The format depends on the language setting on the HMI device.
"Character string"	Input and output of character strings.

Note

Data formats

Not all data formats are available for selection for Runtime Professional.

Hidden input

In Runtime the input can be displayed normally or encrypted, for example for hidden input of a password. A "*" is displayed for every character during hidden input. The data format of the value entered cannot be recognized.

1. In the Inspector window, select "Properties > Properties > Response":
2. Activate "Hidden input".

Avoid overlaps in output fields

If several I/O fields are configured as output fields with a transparent background in a screen, these I/O fields may overlap. The transparent part of the one field covers the digits of the other field. This may cause display problems in Runtime. In order to avoid such overlaps, set the margins of the I/O fields to zero in the object properties under "Properties > Properties > Appearance". Activate "Properties > Properties > Layout > Fit object to contents."

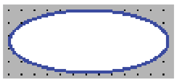
See also

Device-Specific Nature of the Objects (Page 2083)

Ellipse

Application

The "Ellipse" is an enclosed object that can be filled with a color or pattern.



Layout

In the Inspector window you can customize the settings for the object position, geometry, style, frame and color. You can adapt the following properties in particular:

- Horizontal radius: Specifies the horizontal radius of the elliptical object.
- Vertical radius: Specifies the vertical radius of the elliptical object.

Horizontal radius

The horizontal radius of the "Ellipse" object is specified in the Inspector window. The value is entered in pixels.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value between 0 and 2500 under "Horizontal."

Vertical radius

The vertical radius of the "Ellipse" object is specified in the Inspector window. The value is entered in pixels.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value between 0 and 2500 at "Vertical."

See also

Device-Specific Nature of the Objects (Page 2083)

Graphic view

Application

The "Graphic view" object is used to display graphics.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Graphic: Specifies the graphic file that is displayed in the object.
- Stretch graphic: Specifies the automatic size stretching for objects with graphics.
- Transparent color: Specify whether or not the transparent color is used for the graphic.

Inserting graphics

The following graphic format is used in the "Graphic view" object: *.bmp, *.tif, *.png, *.ico, *.emf, *.wmf, *.gif, *.jpg or *.jpeg. You may also use graphics as OLE objects in the Graphic view .

1. In the Inspector window, select "Properties > Properties > General":
2. Select the graphic that you wish to insert.

   The graphic preview is shown in the right pane.
3. Click "Apply" to insert the graphic in the Graphic view .

Stretch graphic

Whether a graphic displayed in a Graphic view is stretched to the size of the Graphic view in runtime is specified in the Inspector window.

1. Click "Properties > Properties > Layout" in the inspector window.
2. Select the required size adjustment for the graphic.

Transparent color

This property defines whether the transparent color is used for the graphic to be displayed.

1. Click "Properties > Properties > Appearance" in the inspector window:
2. Activate "Background > Transparent".
3. Select a transparent color.

Note

When using bitmaps in WinCC screens the "Transparent color" setting demands a high character performance in the layout on the panel. Visualization performance is enhanced by disabling the "Transparent" setting in the properties of the relevant display object. This restriction applies in particular when bitmaps are used as background image.

Note

Basic Panels

The "Transparent" property is not available for Basic Panels.

See also

Device-Specific Nature of the Objects (Page 2083)

Storing an external image in the graphics library. (Page 2007)

Options for Editing Objects (Page 1986)

Objects for Basic Panels (Page 2083)

Graphic I/O field

Application

The "Graphic I/O field" object can be used to configure a list for display and selection of graphic files.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Mode: Specifies the response of the object in Runtime.
- Scroll bar type: Specifies the graphic layout of the scroll bar.

Note

Basic Panels

The scroll bar is not available for Basic Panels.

Note

Reports

Graphic I/O fields output exclusively graphics in reports. "Output" mode is preset. Properties for configuring the selection of graphics are not available, e.g. "scroll bar".

Mode

The response of the "Graphic I/O field" object is specified under "Properties > Properties > General > Type > Mode" in the Inspector window.

Mode	Description
"Input"	The "Graphic I/O field" object is only used to select graphics.
"Input/output"	The "Graphic I/O field" object is used to select and display graphics.
"Output"	The "Graphic I/O field" object is used to display graphics only.
"Two states"	The "Graphic I/O field" object is only used to display graphics and can have a maximum of two states. You use no graphics list but insert one graphic each for the "ON" and "OFF" state.

Stretch graphic

Whether a graphic displayed in a graphic I/O field is stretched to the size of the view in runtime is specified in the Inspector window.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Select the required size adjustment for the graphic.

Scroll bar type

The response for the graphic representation of the scroll bar is specified under "Properties > Properties > Appearance > Scroll Bar > Type" in the Inspector window.

Type	Description
"Permanent"	The scroll bar is always visible.
"No scrollbar"	The scroll bar is not visible.
"Visible after clicking"	The scroll bar is made visible by a mouse click.

See also

Device-Specific Nature of the Objects (Page 2083)

Symbolic I/O field (Page 2108)

Help indicator (Basic, Advanced)

Application

The object "help indicator" is available for the HMI devices OP 73 and KP300 Basic. If a a tooltip exists for the selected object, the help indicator is displayed during runtime. If a tooltip was configured for the opened screen, the help indicator always remains visible.

You configure the object "help indicator" exclusively in the global screen.

Layout

You can adapt the following properties in the Inspector window:

- Position: Determines the position of the object "Help indicator."

Position

You can use this property to set the position of the object "Help indicator."

1. Select the object "Help indicator" in the template.
2. In the Inspector window, select "Properties > Properties > Layout".
3. Enter a value for X and Y. You can also use the cursor keys to position the selected object.

If you have configured a screen object at this position, the visible help indicator covers the screen object. The help indicator is covered only by incoming system alarms and dialogs.

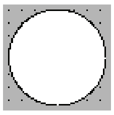
See also

Device-Specific Nature of the Objects (Page 2083)

Circle

Application

The "Circle" object is a closed object which can be filled with a color or pattern.



Layout

In the Inspector window you can customize the settings for the object position, geometry, style, frame and color. You can adapt the following properties in particular:

- Radius: Specifies the size of the circle.

Radius

The radius of the "Circle" object is specified in the Inspector window. The value is entered in pixels.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value between 0 and 2500 in the "Radius" area.

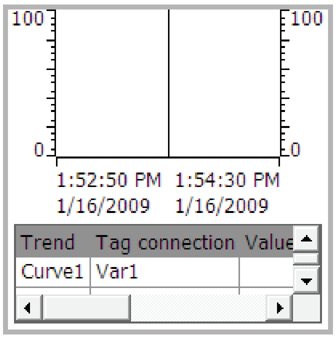
See also

Device-Specific Nature of the Objects (Page 2083)

Trend view

Application

The trend view is meant for the graphical representation of tag values from the current process or from a log in form of trends.



Trend	Tag connection	Value
Curve1	Var1	

Layout

In the Inspector window, you customize the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Display value table, ruler and grid: Specifies whether a value table, a ruler or a grid is displayed in addition to the coordinate system to improve legibility.
- Toolbars: Defines the display of the control elements.

Display value table, ruler and grid

For improved legibility a value table, a ruler and a grid can be displayed in Runtime.

1. Activate "Properties > Properties > Appearance > Show ruler".
2. Activate "Properties > Properties > Table > Show table".
3. Activate "Properties > Properties > Table > Show grid".

Toolbars

The layout of the control elements is defined in the "Properties > Properties > Toolbar" inspector window.

Note

Basic Panels

As archiving is not possible for Basic Panels, the control elements are not available.

Toolbar button	Brief description	Description
⏮	"Go to start"	Scrolls back to the beginning of the trend recording. The start values with which the trend recording started are displayed.
⊕	"Zoom in"	Zooms into the displayed time section.
⊖	"Zoom out"	Zooms out of the displayed time section.
⇐	"Ruler backward"	Moves the ruler back.
⇒	"Ruler forward"	Moves the ruler forward.
⏪	"Backward"	Scrolls back one display width.
⏩	"Forward"	Scrolls forward one display width.
▥	"Ruler"	Shows or hides the ruler. The ruler displays the X-value associated with a Y-value.
▶ ■	"Start/stop"	Stops trend recording or continues trend recording.

Configuration behavior

Displaying column headers

The layout of the table in the trend view depends on the view settings in the Control Panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display > Appearance" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in Runtime.

Consistency test

If warnings or errors are displayed in the output window during a consistency check in connection with trend views, clicking "Go to Error/Tag" on the shortcut menu will not always take you to the exact error position. In some cases only the trend view is shown as cause of error.

Adding, configuring, and removing trends

The trends of the trend view are managed in the Inspector window under "Properties > Properties > Trend." You can copy trends between different trend views.

See also

Device-Specific Nature of the Objects (Page 2083)

Configuring trend displays for values from the PLC (Page 2155)

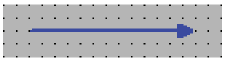
Touch and key operation (Page 2816)

Overview (Page 2815)

Line

Application

The "Line" object is an open object. The line length and gradient slope are defined by the height and width of the rectangle enclosing the object.



Layout

In the Inspector window, you customize the settings for the object position, shape, style, and color. You can adapt the following properties in particular:

- Line style
- Line start and end

Line style

The representation of the line is specified under "Properties > Properties > Appearance" in the Inspector window. The line is shown without interruption if you select "Solid", for example.

Note

The line styles available depend on the selected HMI device.

Line start and end

The start and end points of the line are specified under "Properties > Properties > Appearance > Line ends" in the Inspector window.

Use arrow point, for example, as start and end point. The available start and end points depend on the device.

See also

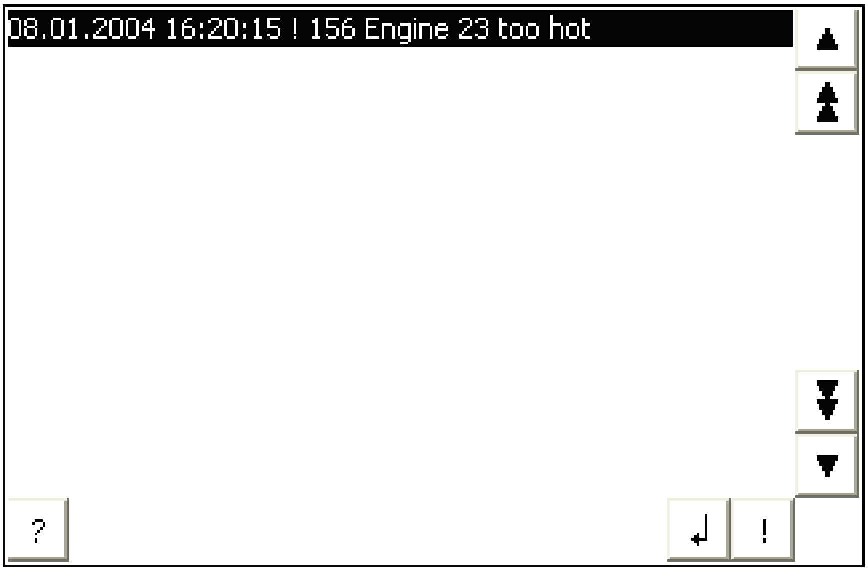
Device-Specific Nature of the Objects (Page 2083)

Alarm view

Application

Alarms are indicated in the Alarm view or in the Alarm window of the HMI device.

The following screen contains a simple alarm view:



08.01.2004 16:20:15 ! 156 Engine 23 too hot

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object.

Note

The fonts available for selection depend on the "Language and fonts" you have configured in the Runtime settings.

You can adapt the following properties in particular:

- Control elements: Defines the operator controls of the alarm display.
- Alarm classes: This setting defines which alarm classes are displayed in the alarm view.
- Columns: Specifies the displayed columns in runtime.

  Note

  If you have different alarm classes output, these will be initially sorted into alarm classes in runtime, and then by when the alarm occurred.

Control elements

The control elements that can be used to control the alarm display in runtime are specified in the Inspector window under "Display > Settings". The following table shows the control elements in the alarm view, and what they do:

Button		Function
"Info text"	?	Displays info text for an alarm.
"Acknowledge"	!	Acknowledges an alarm.
"Loop-In-Alarm"	↲	Switches to the screen containing information about the error that has occurred.

Select alarm classes

1. Click "Properties" in the Inspector window.
2. Under "Alarm classes" activate the alarm classes to be displayed in the alarm view in runtime .

Define columns

Define the columns to be displayed in the alarm view in runtime in the Inspector window.

1. In the Inspector window, click "Properties > Columns".
2. Activate the columns that are to be displayed in runtime under "Columns".

Displaying column headers

The layout of the alarm view is dependent on the view settings in the control panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display > Layout tab" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in runtime.

Note

In the engineering system you can dynamically control the visibility of an object, for example, in the "Animations" group of the Inspector window. In runtime, the "Simple alarm view" does not support animations. If you have configured an animation and, for example, wish to perform a consistency check of the project, then an error alarm is issued in the Output window.

See also

Device-Specific Nature of the Objects (Page 2083)

Alarm window (Page 2100)

Alarm indicator (Page 2102)

Alarm window

Application

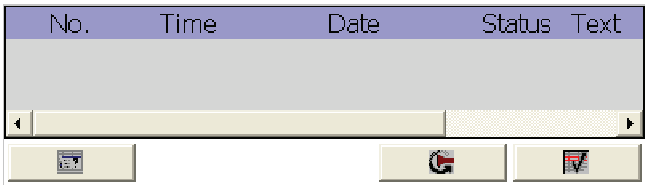
Alarms are indicated in the Alarm view or in the Alarm window of the HMI device. The layout and operation of the Alarm window are similar to that of the Alarm view. The Alarm window has the following characteristics that are the same as in the Alarm view:

- Simple alarm window
- Advanced alarm window
- Alarm line

> **Note**
>
> **Basic Panels**
>
> Only the simple alarm window is available for Basic Panels.

The Alarm window is configured in the "Global screen" editor.

The Alarm window is independent of the process screen. Depending on the configuration, the Alarm window opens automatically as soon as a new, unacknowledged alarm has been received. If applicable, the Alarm window is configured so that it only closes after all alarms have been acknowledged. The following figure shows an advanced Alarm window:



> **Note**
>
> In the engineering system you dynamize, for example, the visibility of an object in "Properties > Animations" in the Inspector window. In runtime, the "Simple alarm window" object does not support animations. If you have configured an animation and, for example, wish to perform a consistency check of the project, then an error alarm is issued in the Output window.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You configure the Alarm window in the same way as the Alarm view. In addition you adapt the following properties:

- Fixed alarm windows: Specifies that the Alarm window retains the focus after a screen change.
- Window: You define the operator input and response of the Alarm window in runtime.

Note

If you have different alarm classes output, these will be initially sorted into alarm classes in runtime, and then by when the alarm occurred.

Control elements

The control elements that can be used to control the alarm view in runtime are specified in the Inspector window under "Properties >Display > Settings". The following table shows the control elements in the Alarm window, and what they do:

Button		Function
"Tooltip"	?	Displays a tooltip for an alarm.
"Acknowledge"	!	Acknowledges an alarm.
"Loop-In-Alarm"	↲	Switches to the screen containing information about the error that has occurred.

Access protection in runtime

Configure access protection under "Properties > Properties > Security" in the Inspector window of the alarm view. If a logged-on user has the required authorization, he can acknowledge and edit alarms using the operator controls in the alarm view. If the logged-in user does not have the required authorization, or if no user is logged in, clicking the "Acknowledge" or "Edit" buttons or double-clicking an alarm line opens the login dialog box.

Note

Basic Panels

Access protection is not available for Basic Panels.

Activating the focus of the Alarm window

Select the following option so that the Alarm window does not lose the focus after a screen change:

1. In the Inspector window, select "Properties > Properties > Mode":
2. Enable "Label".

Window

Define the response of the Alarm window under "Properties > Properties > Mode > Window" in the Inspector window. The following table shows the possible properties:

Option	Function
Automatic display	The Alarm window is automatically displayed when a system alarm occurs, for example.
Closable	The window closes again after a set time has elapsed. You define the display duration in the alarm settings.
Modal	The Alarm window is linked to a confirmation, such as: Alarm must be acknowledged. If the modal alarm window has the focus, the buttons in the screen behind it cannot be used. The functions configured for a function key are carried out.
Sizeable	You can change the size of the Alarm window in runtime.

See also

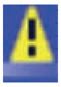
Device-Specific Nature of the Objects (Page 2083)

Alarm view (Page 2098)

Alarm indicator

Application

The alarm indicator is a graphic symbol that shows current errors or errors that need to be acknowledged, depending on the configuration. The alarm indicator is configured in the "Global screen" editor. The following figure shows an alarm indicator:



Alarm indicator OP 73

A "simple" alarm indicator is available for the HMI OP 73. The following diagram shows the alarm indicator for the OP 73 HMI devices:

The "simple" alarm indicator shows alarms to be acknowledged or alarms which have already been acknowledged and have not yet gone. Only the position can be defined for the "simple" alarm indicator. The alarm indicator is displayed on the device at the selected position. If you have configured a screen object at this position, the visible alarm indicator covers the screen object. The alarm indicator is covered by system dialogs, such as the login dialog, Help dialog, and alarm windows.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Alarm classes: Establishes the alarm classes where the alarm indicator is displayed.
- Operator control in Runtime: Defines the operator actions in Runtime that cause the Alarm window to open.

Alarm classes

You define which alarm classes are shown with an alarm indicator in "General > Alarm classes" in the Inspector window. Alarm classes, such as "Warnings" or "Errors".

Define operator control in Runtime

1. Select the alarm indicator in the screen.
2. Click "Events > Click" or "Click when flashing" in the Inspector window.
3. The "function list" opens. Click the first line of the function list. The list of system functions, and scripts available in the project opens.
4. Select the "ShowAlarmWindow". system function under "Alarms."
5. Under "object name" select the name of the Alarm window from the selection list. Under "Layout", define whether the Alarm window should be visible, hidden, or should toggle between the two states.

See also

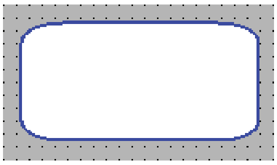
Device-Specific Nature of the Objects (Page 2083)

Alarm view (Page 2098)

Rectangle

Application

The "Rectangle" is a closed object which you can fill with a color.



Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Corner radius: Specifies the horizontal and vertical distance between the corner of the rectangle and the start point of a rounded corner.

Corner radius

The corners of the "Rectangle" object can be rounded to suit your requirements. When the properties "X" and "Y" are set to the 100 % value, the rectangle is displayed as an ellipse. As soon as one of the properties has the value 0%, a normal rectangle without a rounded corner is shown.

1. Click "Properties > Properties > Layout" in the inspector window.
2. Enter a value for "X" in the "Corner radius" area.

   The input value is the percentage proportion of half the width of the rectangle.
3. Enter a value for "Y" in the "Corner radius" area.

   The input value is the percentage proportion of half the height of the rectangle.

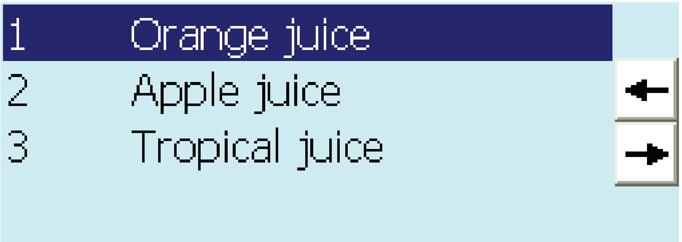
See also

Device-Specific Nature of the Objects (Page 2083)

Recipe view

Application

The "Recipe view" object is used to display and modify recipes.



1 Orange juice
2 Apple juice
3 Tropical juice

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You can adapt the following properties in particular:

- Control elements: Specifies the menu commands of the recipe view.

Control elements

The menu commands with which the recipe view is operated in Runtime are configured under "Properties > Buttons" in the inspector window.

Menu command	Description
"Tooltip"	Calls up the configured tooltip for the selected recipe.
"New record"	Creates a new recipe record in the recipe.
"Delete record"	Deletes the selected record.
"Saving"	Saves the modified record with its current name.
"Save as"	Saves the modified record with a new name.
"Write to PLC"	Sends the current value to the PLC.
"Read from PLC"	Reads the current value from the PLC.

See also

Device-Specific Nature of the Objects (Page 2083)

Simple recipe view (Page 2228)

Displaying recipes (Page 2223)

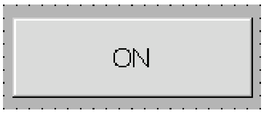
Configuring the simple recipe view (Page 2239)

Switch

Application

The "Switch" object is used to configure a switch that is used to switch between two predefined states in Runtime. The current state of the "Switch" object can be visualized with either a label or a graphic.

The following figure shows a "Switch" type switch.



Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. In particular, you can customize the following property:

- Type: Defines the graphic representation of the object.

Type

The button display is defined in "Properties > Properties > General >Settings" in the Inspector window.

Type	Description
"Switch"	The two states of the "Switch" are displayed in the form of a switch. The position of the switch indicates the current state. The state is changed in runtime by sliding the switch. You specify the direction of movement of the switch in "Switch orientation" with this type.
"Switch with text"	The switch is shown as a button. The current state is visualized with a label. In runtime click on the button to actuate the switch.
"Switch with graphic"	The switch is shown as a button. The current state is visualized with a graphic. In runtime click on the button to actuate the switch.

Note

Basic Panels

The "Switch" type is not available for Basic Panels.

See also

Device-Specific Nature of the Objects (Page 2083)

Elements (Page 2357)

Overview of objects (Page 1983)

Button

Application

The "Button" object allows you to configure an object that the operator can use in runtime to execute any configurable function.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Mode: Defines the graphic representation of the object.
- Text / Graphic: Defines whether the Graphic view is static or dynamic.
- Define hotkey: Defines a key, or shortcut that the operator can use to actuate the button.

Note

You can only define a hotkey for HMI devices with keys.

Mode

The button display is defined in "Properties > Properties > General >Mode" in the Inspector window.

Mode	Description
"Invisible"	The button is not visible in runtime.
"Text"	The current state of the button is visualized with a label.
"Graphic"	The current state of the button is visualized with a graphic.

Depending on the device, further options are available:

Text / Graphic

The "Mode" property settings are used to define whether the display is static or dynamic. The display is defined in "Properties > Properties > General >Label > Text" or "Graphic" in the Inspector window.

Type	Option	Description
"Text"	"Text"	"Text OFF" is used to specify a text that appears in the button when the state is "OFF". If you enable "Text ON", you can enter a text for the "ON" state.
	"Text list"	The text in the button depends on the state. The entry from the text list corresponding to the state is displayed.
"Graphic"	"Graphic"	"Graphic OFF" is used to specify a graphic that is displayed in the button when the state is "OFF". If you enable "Graphic ON", you can enter a graphic for the "ON" state.
	"Graphics list"	The graphic in the button depends on the state. The entry from the graphics list corresponding to the state is displayed.

Define hotkey

In the Inspector window, a key or key combination is defined that the operator can use to control the button in runtime.

1. In the Inspector window, select "Properties > Properties > General":
2. Select a key or key combination from the selection list in the "Hotkey" area.

See also

Device-Specific Nature of the Objects (Page 2083)

Symbolic I/O field

Application

The "Symbolic I/O field" object can be used to configure a selection list for input and output of texts in Runtime.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Mode: Specifies the response of the object in Runtime.
- Text list: Specifies the text list that is linked to the object.
- Button for selection list: Specifies that the object has a button to open the selection list.

Note

Reports

In reports, symbolic I/O fields only output data. "Output" mode is preset. Properties for configuring the selection of graphics are not available, e.g. "button for selection list".

Mode

The response of the symbolic I/O field is specified in the Inspector window in "Properties > Properties > General > Type".

Mode	Description
"Output"	The symbolic I/O field is used to output values.
"Input"	The symbolic I/O field is used to input values.
"Input/output"	The symbolic I/O field is used for the input and output of values.
"Two states"	The symbolic I/O field is used only to output values and has a maximum of two states. The field switches between two predefined texts. This is used, for example, to visualize the two states of a valve: closed or open.

Note

The behavior possible for the symbolic I/O field depends on the Runtime.

Text list

In the Inspector window, you specify which text list is linked to the symbolic I/O field.

1. In the Inspector window, select "Properties > Properties > General":
2. Under "Contents" open the selection list for "Text list".
3. Select a text list.

Button for selection list

The "Button for selection list" property is used to display a button for opening the selection list.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Activate the "Button for selection list" option under "Response".

> **Note**
>
> **Basic Panels**
>
> The "Button for selection list" option is not available for Basic Panels.

See also

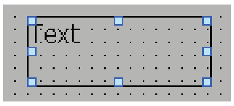
Device-Specific Nature of the Objects (Page 2083)

Graphic I/O field (Page 2092)

Text field

Application

The "Text field" is a closed object which you can fill with a color.



Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Text: Specifies the text for the text field.
- Size of text field: Defines whether the size of the object is adapted to the space required by the largest list entry.

Text

Specify the text for the text field in the Inspector window.

1. In the Inspector window, select "Properties > Properties > General":
2. Enter a text.

   For texts over several lines you can set a line break by pressing the key combination <Shift + Enter>.

Size of text field

In the Inspector window, you can define whether the size of the object is adapted to the space required by the largest list entry.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Activate "Auto-size".

Otherwise you can set the size manually. Deactivate "Auto-size". Press the mouse button and drag the text field to the required size.

See also

Device-Specific Nature of the Objects (Page 2083)

10.1.9 Configuring screen navigation

10.1.9.1 Basics for screen navigation

Types of navigation for the screen change

For a production process consisting of multiple subprocesses, you will configure multiple screens. You have the following options to enable the operator to switch from one screen to the next in Runtime:

- Assign buttons to screen changes
- Configuring screen changes at local function keys

Procedure

Before you create a screen change, define the plant structure and derive from it the screen changes that you want to configure.

Create the start screen under "Runtime Settings > General > Start screen".

See also

Assign button with screen change (Page 2112)

10.1.9.2 Assign button with screen change

Introduction

Configure a button in the screen to switch between the screens on the HMI device during operation.

Note

If you have set the "Visibility" of animations to "Hidden" in the Inspector window of a screen, this screen cannot be called up in Runtime.

Requirements

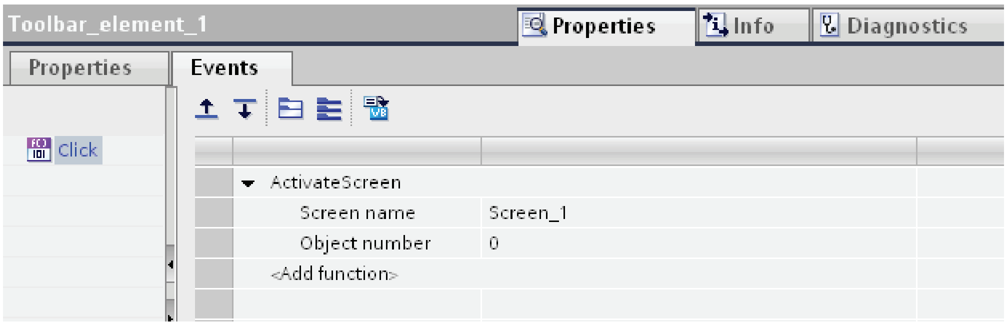
- You have created the project.
- You have created the "Screen_2" screen.
- "Screen_1" is created.

Procedure

1. Double-click "Screen_1" in the project navigation. The screen is displayed in the work area.
2. Move "Screen_2" from the project tree to the open screen by drag&drop.

   A button with the name "Screen_1" is inserted.
3. In the Inspector window, select "Properties > Events > Click".

   The "ActivateScreen" system function is displayed in the "Function list".

   ActivateScreen
   Screen name Screen_1
   Object number 0
   <Add function>
4. At the "Object number" attribute, define, if required, the tab sequence number of the object on which the focus is to be set after a screen change. You can also specify a tag that contains the object number.

Alternative procedure

1. Move a button from the "Tools" task card to "Screen2" by drag&drop.
2. In the Inspector window, select "Properties > Events > Click".
3. Select the "ActivateScreen" system function.
4. Select "Screen_2" for the "Screen number".

Result

The operator goes to "Screen_1 with the button in Runtime. If you have specified an object number, the object with this object number has the focus following a screen change.

See also

Basics for screen navigation (Page 2111)

10.1.9.3 Assign screen change to function key

Introduction

Configure a screen change function key in the screen to switch between the screens on the HMI device during operation.

Note

If you have set the "Visibility" of animations to "Hidden" in the inspector window of a screen, this screen cannot be called up in Runtime.

Requirements

- You have created a project.
- You have created the "Screen_2" screen.
- You have created the "Screen_1" screen.

Procedure

1. Double click "Screen_1" in the project tree. The screen is displayed in the work area.
2. Move "Screen_2" from the project tree to a function key, e.g. "F2".

   The configured function key displays a yellow triangle.
3. Click "Properties > Events > Press key" in the inspector window.

   The "ActivateScreen" system function is displayed.

Result

The operator goes to the specified "Screen_2" with function key "F2" in Runtime.

10.2 Working with Tags

10.2.1 Basics

10.2.1.1 Basics of tags

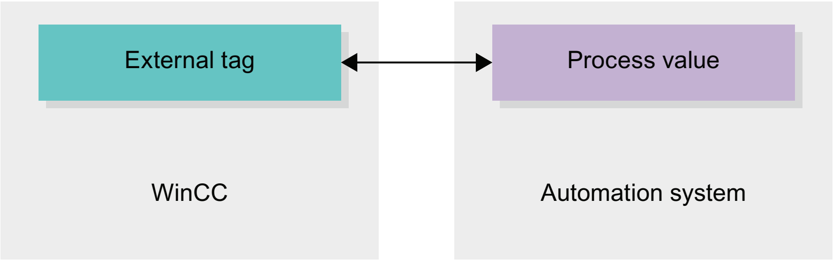
Introduction

Process values are forwarded in runtime using tags. Process values are data which is stored in the memory of one of the connected automation systems. They represent the status of a plant in the form of temperatures, fill levels or switching states, for example. Define external tags for processing the process values in WinCC.

WinCC works with two types of tag:

- External tags
- Internal tags

The external tags form the link between WinCC and the automation systems. The values of external tags correspond to the process values from the memory of an automation system. The value of an external tag is determined by reading the process value from the memory of the automation system. It is also possible to rewrite a process value in the memory of the automation system.



Internal tags do not have a process link and only convey values within the WinCC.

Tags in WinCC

For external tags, the properties of the tag are used to define the connection that the WinCC uses to communicate with the automation system and form of data exchange.

Tags that are not supplied with values by the process - the internal tags - are not connected to the automation system. In the tag's "Connection" property, this is identified by the "Internal tag" entry.

You can create tags in different tag tables for greater clarity. You then directly access the individual tag tables in the "HMI tags" node in the project tree. The tags from all tag tables can be displayed with the help of the table "Show all tags".

See also

Overview of HMI tag tables (Page 2116)

Internal tags (Page 2121)

External tags (Page 2117)

Addressing external tags (Page 2118)

Creating external tags (Page 2122)

Basics on arrays (Page 2150)

Cycle basics (Page 2154)

10.2.1.2 Overview of HMI tag tables

Introduction

HMI tag tables contain the definitions of the HMI tags that apply across all devices. A tag table is created automatically for each HMI device created in the project.

In the project tree there is an "HMI tags" folder for each HMI device. The following tables can be contained in this folder:

- Standard tag table
- User-defined tag tables
- All tags

The following tables are also available in an HMI tag table:

- Discrete alarms
- Analog alarms

With the help of these tables you configure alarms for the currently selected HMI tag.

In the project tree you can create additional tag tables in the HMI tags folder and use these to sort and group tags and constants. You can move tags to a different tag table using a drag-and-drop operation or with the help of the "Tag table" field. You activate the "Tag table" field using the shortcut menu of the column headings.

Standard tag table

There is one standard tag table for each HMI device of the project. It cannot be deleted, renamed or moved. The standard tag table contains HMI tags and, depending on the HMI device, also system tags. You can declare all HMI tags in the standard tag table, or create additional user-defined tag tables as you want.

User-defined tag tables

You can create multiple user-defined tag tables for each HMI device in order to group tags according to your requirements. You can rename, gather into groups, or delete user-defined tag tables. To group tag tables, create additional subfolders in the HMI tags folder.

All tags

The "All tags" table shows an overview of all HMI tags and system tags of the HMI device in question. This table cannot be deleted, renamed or moved.

Discrete alarms table

In the "Discrete alarms" table, you configure discrete alarms to the HMI tag selected in the HMI tag table. When you configure a discrete alarm, multiple selection in the HMI tag table is not possible. You configure the discrete alarms for each HMI tag separately.

Analog alarms table

In the "Analog alarms" table, you configure analog alarms to the HMI tag selected in the HMI tag table. When you configure an analog alarm, multiple selection in the HMI tag table is not possible. You configure the analog alarms for each HMI tag separately.

See also

Basics of tags (Page 2115)

10.2.1.3 External tags

Introduction

External tags allow communication (exchange of data) between the components of an automation system, such as between the HMI device and the PLC.

Principle

An external tag is the image of a defined memory location in the PLC. You have read and write access to this storage location from both the HMI device and from the PLC.

Since external tags are the image of a storage location in the PLC, the applicable data types depend on the PLC which is connected to the HMI device.

In STEP 7, if you write a PLC control program, the PLC tags created in the control program will be added to the PLC tag table. If you want to connect an external tag to a PLC tag, access the PLC tags directly via the PLC tag table and connect them to the external tag.

Data types

All the data types which are available at the connected PLC are available at an external tag in WinCC. Information about data types which are available for connection to other PLCs can be found in the documentation about the respective communication drivers.

See "Basics of communication (Page 2381)" for additional information.

Note

As well as external tags, area pointers are also available for communication between the HMI device and PLC. You can set up and enable the area indicators in the "Connections" editor.

Update of tag values

For external tags, the current tag values are transmitted in runtime via the communication connection between WinCC and the connected automation systems and then saved in the runtime memory. Next, the tag value will be updated to the set cycle time. For use in the runtime project, WinCC accesses tag values in the runtime memory that were read from the PLC at the previous cycle time. As a result, the value in the PLC can already change whilst the value from the runtime memory is being processed.

See also

Addressing external tags (Page 2118)

Basics of communication (Page 2381)

Basics of tags (Page 2115)

10.2.1.4 Addressing external tags

Introduction

The options for addressing external tags depend on the type of connection between WinCC and the PLC in question. A distinction must be made between the following connection types:

- Integrated connection
  Connections of devices which are within a project and were created with "Devices & Networks" editor are referred to as integrated connections.
- Non-integrated connection
  Connections of devices which were created with the "Connections" editor are referred to as non-integrated connections. It is not necessary that all of the devices be within a single project.

The connection type can also be recognized by its symbol.

🖧	Integrated connection
↳	Non-integrated connection

You can find additional details on this in the section "Basics of communication (Page 2381)".

Addressing with integrated connections

An integrated connection offers the advantage that you can address a tag both symbolically and absolutely.

For symbolic addressing, you select the PLC tag via its name and connect it to the HMI tag. The valid data type for the HMI tag is automatically selected by the system. You have to distinguish between the following cases when you address elements in data blocks:

- Symbolic addressing of data blocks with optimized access
  For symbolic addressing of a data block with optimized access, the address of an element in the data block is assigned dynamically and automatically applied to the HMI tag in case of a change. You do not need to compile the connected data block or the WinCC project for this step.
  For data blocks with optimized access, only symbolic addressing is available.
- Symbolic addressing of data blocks with standard access
  For symbolic addressing of a data block with standard access, the address of an element in the data block is assigned permanently. The valid data type for the HMI tag is automatically selected by the system. Any change in the address of an element in the data block is applied directly to the HMI tag. You do not need to compile the connected data block or the WinCC project for this step.
  For data blocks with standard access, you can use symbolic addressing as well as absolute addressing.

For symbolic addressing of elements in a data block, you only need to recompile and reload the WinCC project in case of the following changes:

- If the name or the data type of the connected data block has changed.
- If the name or the data type of a higher-level structure node of the connected element in the data block has changed.
- If the name of the connected data block has changed.

Symbolic addressing is currently only available on PLCs of the SIMATIC S7 1200 type. Addressing with optimized access is only available in an integrated connection.

You can also use absolute addressing with an integrated connection. You have to use absolute addressing for PLC tags from a SIMATIC S7 300/400 PLC. If you have connected an HMI tag with a PLC tag and the address of the PLC tag changes, you only have to recompile the control program to update the new address in WinCC. Then you recompile the WinCC project and load it onto the HMI device.

In WinCC, symbolic addressing is the default method. To change the default setting, select the menu command "Options > Settings". Select "Visualization > Tags" in the "Settings" dialog. If required, disable the "Symbolic access" option.

The availability of an integrated connection depends on the PLC used. The following table shows the availability:

PLC	Integrated connection	Comments
S7 300/400	Yes	The linking of tags is not checked in Runtime. If the tag address changes in the PLC and the HMI device is not compiled again and loaded, the change is not registered in runtime.
S7 1200	Yes	A validity check of the tag connection is performed in runtime during symbolic addressing. If an address is changed in the PLC, the change is registered and an error message is issued. The reaction described for S7 300/400 applies whilst addressing with standard access.

Create an integrated connection in the "Devices & Networks" editor. If the PLC is contained in the project and integrated connections are supported, you can then also have the connection created automatically. To do this, when configuring the HMI tag, simply select an existing PLC tag to which you want to connect the HMI tag. The integrated connection is then automatically created by the system.

Addressing with non-integrated connections

In the case of a project with a non-integrated connection, you always configure a tag connection with absolute addressing. Select the valid data type yourself. If the address of a PLC tag changes in a project with a non-integrated connection during the course of the project, you also have to make the change in WinCC. The tag connection cannot be checked for validity in Runtime, an error message is not issued.

A non-integrated connection is available for all supported PLCs.

Symbolic addressing is not available in a non-integrated connection.

With a non-integrated connection, the control program does not need to be part of the WinCC project. You can perform the configuration of the PLC and the WinCC project independently of each other. For configuration in WinCC, only the addresses used in the PLC and their function have to be known.

See also

External tags (Page 2117)

Basics of tags (Page 2115)

Basics of communication (Page 2381)

10.2.1.5 Internal tags

Introduction

Internal tags do not have any connection to the PLC.

Principle

Internal tags are stored in the memory of the HMI device. Therefore, only this HMI device has read and write access to the internal tags. You can create internal tags to perform local calculations, for example.

You can use the HMI data types for internal tags.

The following HMI data types are available:

HMI data type	Data format
Array	One-dimensional array
Bool	Binary tag
DateTime	Date/time format
DInt	Signed 32-bit value
Int	Signed 16-bit value
LReal	Floating-point number 64-bit IEEE 754
Real	Floating-point number 32-bit IEEE 754
SInt	Signed 8-bit value
UDInt	Unsigned 32-bit value
UInt	Unsigned 16-bit value
USInt	Unsigned 8-bit value
WString	Text tag, 16-bit character set

See also

Basics of tags (Page 2115)

10.2.2 Working with tags

10.2.2.1 Creating tags

Creating external tags

Introduction

You can access an address in the PLC via a PLC tag using an external tag. The following options are available for addressing:

- Symbolic addressing
- Absolute Addressing

For additional information on symbolic addressing see the section "Addressing external tags (Page 2118)". If possible, use symbolic addressing when configuring a tag. You create tags either in the standard tag table or in a tag table you defined yourself.

Requirement

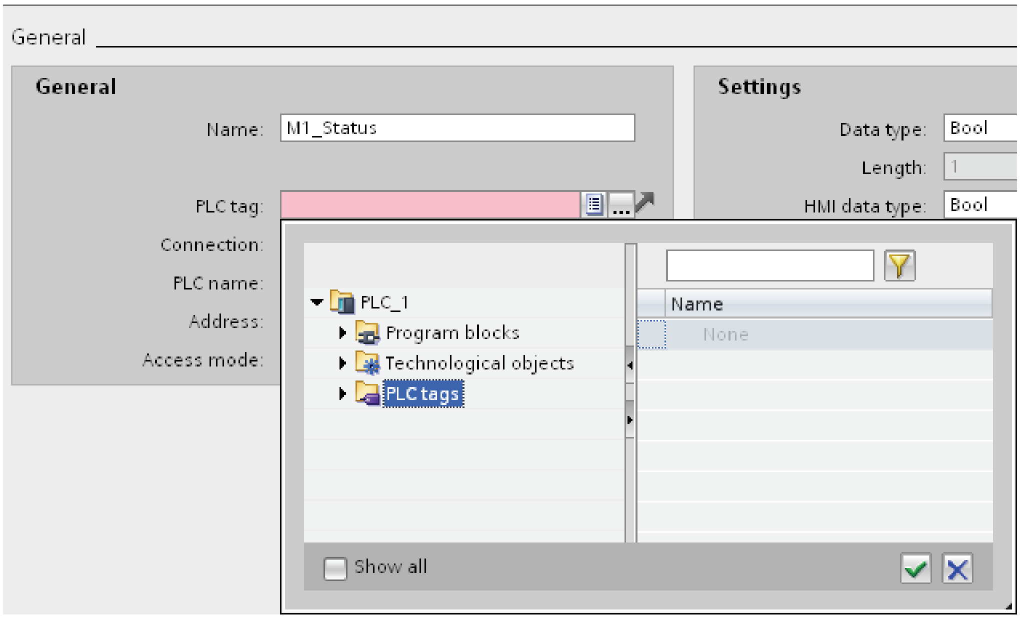
- You have opened the project.
- A connection to the PLC is configured.
- The Inspector window is open.

Procedure

To create an external tag, proceed as follows:

1. Open the "HMI tags" folder in the project tree and double-click the standard tag table. The tag table is opened.
   Alternatively, create a new tag table and then open it.
2. In the "Name" column, double-click "Add" in the tag table. A new tag is created.
3. Select the "Properties > Properties >General" category in the Inspector window and, if required, enter a unique tag name in the "Name" field. The tag name must be unique throughout the device.
4. If required, select the "Display name" field to enter a name to be displayed in Runtime. The name to be displayed is language-specific and can be translated for the required Runtime languages. The display name is available for Basic Panels, Panels and Runtime Advanced.

5. Select the connection to the required PLC in the "Connection" box. If the connection you require is not displayed, you must first create the connection to the PLC. You create the connection to a SIMATIC S7 PLC in the "Devices & Networks" editor. You create the connection to external PLCs in the "Connections" editor.
   If the project contains the PLC and supports integrated connections, you can have the connection created automatically. To do this, when configuring the HMI tag, simply select an existing PLC tag to which you want to connect the HMI tag. The integrated connection is then automatically created by the system.
6. If you are working with an integrated connection, click the … button in the "PLC tag" field and select an already created PLC tag in the object list. Click the ✓ button to confirm the selection.



General

Name: M1_Status

Settings

Data type: Bool

Length: 1

HMI data type: Bool

7. If you are working with a non-integrated connection, enter the address from the PLC in the "Address" field. The "PLC tag" field remains empty.
8. Configure the other properties of the tag in the inspector window.

You can also configure all tag properties directly in the tag table. To view hidden columns, activate the column titles using the shortcut menu.

You can also create new tags alternatively directly at the application point, e.g. on an I/O field. by clicking the <Add new> button in the object list. You then configure the new tag in the Inspector window.

Result

An external tag has been created and linked to a PLC tag or an address in the PLC.

See also

Creating internal tags (Page 2124)

Creating multiple tags (Page 2125)

Editing a Tag (Page 2127)

Tag limits (Page 2132)

Basics of tags (Page 2115)

Addressing external tags (Page 2118)

Creating internal tags

Introduction

You must at least set the name and data type for internal tags. Select the "Internal tag" item, rather than a connection to a PLC.

For documentation purposes, it is a good idea to enter a comment for every tag.

Requirement

You have opened the project.

Procedure

1. Open the "HMI tags" folder in the project tree and double-click the entry "Standard tag table". The tag table is opened.
   Alternatively, create a new tag table and then open it.
2. If the Inspector window is not open, select the "Inspector window" option in the "View" menu.
3. Select the "Properties > Properties >General" category in the Inspector window and, if required, enter a unique tag name in the "Name" field. This tag name must be unique throughout the project.
4. If required, select the "Display name" field to enter a name to be displayed in Runtime. The name to be displayed is language-specific and can be translated for the required Runtime languages. The display name is available for Basic Panels, Panels and Runtime Advanced.
5. Select "Internal tag" as the connection in the "Connection" field.
6. Select the required data type in the "Data type" field.
7. In the "Length" field, you must specify the maximum number of characters to be stored in the tag according to the selected data type. The length is automatically defined by the data type for numerical tags.
8. As an option, you can enter a comment regarding the use of the tag. To do so, click "Properties > Properties > Comment" in the Inspector window and enter a text.

You can also configure all tag properties directly in the tag table. To view hidden columns, activate the column titles using the shortcut menu.

You can also create new tags alternatively directly at the application point, e.g. on an I/O field. by clicking the <Add new> button in the object list. You can then configure the new tag in the Properties window that opens.

Result

An internal tag is created. You can now use this in your project.

In additional steps you can configure the tag, for example, by setting the start value and limits.

See also

Creating external tags (Page 2122)

Creating multiple tags

Introduction

In a tag table, you create additional identical tags by automatically filling the rows of the table below a tag.

The tag names are incremented automatically when filling in automatically.

You can also use the auto fill function to fill table cells below a tag with a single tag property and thus modify the corresponding tags.

If you apply the automatic filling in to already filled cells of a tab table, you will be asked whether you want to overwrite the cells or insert new tags.

If you do not want to overwrite already configured tags, activate insert mode. Activate insert mode by keeping the <Ctrl> key pressed during insertion. Already existing entries in the tag table are moved down if insert mode is activated.

Requirement

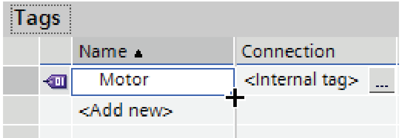
- You have opened the project.
- A tag table is open.
- The tag which is to serve as a template for other tags is configured.

Procedure

1. If you want to create new tags, mark in the "Name" column the tag that should be used as a template for the new tags.

   If you want to copy a property of a tag to the tags below it, select the cell which contains this property.

   The selected cell will be highlighted in color and a small blue square will appear in its bottom right corner. If you move the mouse over this square, the cursor will change to a black cross.

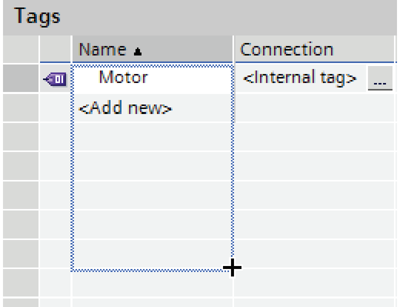


Tags

Name ▲	Connection
Motor	<Internal tag> ...
<Add new>	

2. Hold down the mouse button and drag this square over the cells below that you wish to fill automatically.

   The marking will be extended to cover this area.

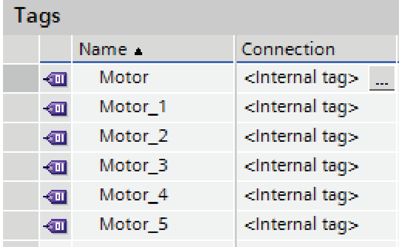


Tags

Name ▲	Connection
Motor	<Internal tag> ...
<Add new>	

3. Now release the mouse button. All of the marked cells will be filled automatically.

   New tags will be created in all empty cells in the marked area.



Tags

Name ▲	Connection
Motor	<Internal tag> ...
Motor_1	<Internal tag>
Motor_2	<Internal tag>
Motor_3	<Internal tag>
Motor_4	<Internal tag>
Motor_5	<Internal tag>

Result

Depending on which cells were selected, the function may automatically fill individual properties or create new tags.

See also

Creating external tags (Page 2122)

10.2.2.2 Editing tags

Editing a Tag

Introduction

You can always rename, copy or delete tags.

When a tag is renamed, the new name must be unique for the whole device.

If you use the "Copy" command to copy a tag to the clipboard, the objects and references linked to the tag are copied as well.

If you use the "Insert" command to add a tag to another device, the tag will be added without the connected references. Only the object name of the reference will be inserted. If a reference of the same name and valid properties exists in the target system, the existing reference will then be connected to the copied tag.

If you copy a tag, the alarms linked to the tag are copied as well. If you add the copied tag to another device, the tag is added together with the linked alarms.

Requirement

- The tag which you wish to rename, copy or delete must exist.
- The tag table is open.

Renaming tags

1. In the "Name" field, select the tag in the tag table.
2. Select "Rename" from the shortcut menu.
3. Type in a new name.

   The tag appears under its new name.

Copying tags

1. Select one or more tags in the tag table or in the Detail window.
2. Select "Copy" from the shortcut menu.
3. Click on the point at which you want to insert the tag. For example, click another tag table in the same device or the tag table in a second device.
4. Select the "Insert" or "Extended insert" command from the shortcut menu. The tag is inserted as described above.

Deleting a tag

1. Select one or more tags in the tag table.
2. Select the "Cross-reference" command from the "Tools" menu. In the "Cross-reference" editor, check to see where the tags are used. In this manner, you can see what impact the deletion of the tag will have on your project.
3. Select "Delete" in the pop-up menu of the tag.

   All marked tags will be deleted.

Export and import of tags

WinCC gives you the option to export and import tags. With Export and Import, you have the option to export tags from one project and import them into another project. There is also the option to create larger numbers of tags outside of WinCC, edit them and subsequently import into any WinCC project. See Importing and exporting tags (Page 2706) for additional information.

See also

Changing the tag configuration (Page 2128)

Configuring multiple tags simultaneously (Page 2129)

Using multiple tags simultaneously in a screen (Page 2130)

Reconnecting a tag (Page 2131)

Creating external tags (Page 2122)

Importing and exporting tags (Page 2706)

Changing the tag configuration

Introduction

You can modify tags at any time to adapt them to changed requirements in the project.

Changing the tag configuration

if you want to change the configuration of a tag, open the tag table in which the tag is contained. Open the "Show all tags" tag table alternatively.

In the tag tables, you can perform such tasks as comparing and adjusting the properties of multiple tags or sorting the tags by their properties.

Change the properties either directly in the table or in the inspector window.

If you change a tag property and this change causes a conflict with another property, it will be highlighted in color to draw your attention to this fact. This could happen, for example, if you connect the tag to another PLC which does not support this data type.

See also

Editing a Tag (Page 2127)

Configuring multiple tags simultaneously

Introduction

In WinCC, you can assign the same properties to multiple tags in a single operation. This facilitates efficient programming.

Requirement

- You created the tags you want to configure.
- The tag table is open.
- The Inspector window is open.

Procedure

1. In the tag table, select all the tags that you want to configure at the same time.

   If the selected property is identical for all the tags, the setting for this property will appear in the Inspector window. The associated field will remain blank otherwise.
2. You can define the shared properties in the Inspector window or directly in the tag table.

if you change a property commonly on several tags, only this one property is changed. The other properties of the tag remain unchanged.

Result

All marked tags will be reconfigured.

To edit tag properties which differ from one tag to the other, simply clear the multiple selection.

See also

Editing a Tag (Page 2127)

Using multiple tags simultaneously in a screen

Introduction

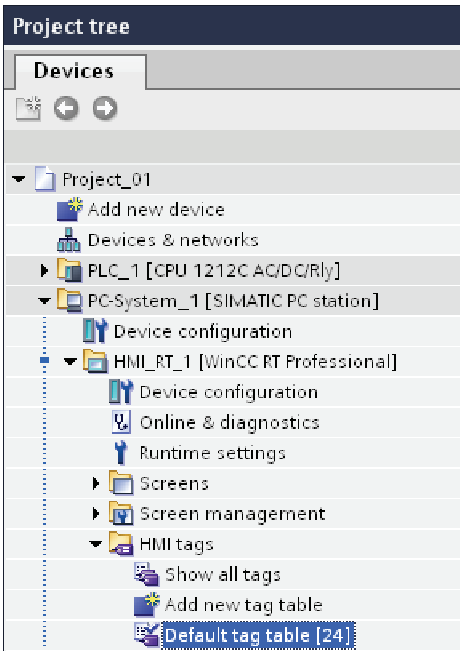
In WinCC, you can create multiple I/O fields that are linked with tags in one screen in a single operation. This facilitates efficient programming.

Requirement

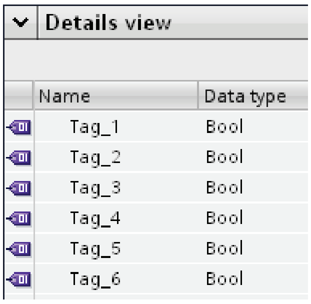
- Several tags are set up.
- A screen is open.

Procedure

1. In the project tree, select the required tag table under "HMI tags".



2. Select the detail view at the bottom of the project tree. The detail view shows the tags that exist in the selected tag group.



Name	Data type
Tag_1	Bool
Tag_2	Bool
Tag_3	Bool
Tag_4	Bool
Tag_5	Bool
Tag_6	Bool

3. Mark the tags in the detail window.
4. Drag the tags to the screen. For each tag, this creates an I/O field that is connected to the tag.

Note

When you move a PLC tag from the detail window to the work area by drag&drop, a network and a connection are created additionally in the "Devices & Networks" editor.

See also

Editing a Tag (Page 2127)

Reconnecting a tag

Introduction

WinCC enables you to automatically connect tags to addresses in the PLC. This procedure is suitable if, for example, changes were made to the connection between the HMI device and the PLC and the tag connections were lost. The function can also be used if you have configured the control program and HMI project separately.

The shortcut menu "Reconnect PLC tag" is available for this.

The menu command is available under the following conditions:

- An integrated connection to the PLC is present.
- The absolute address from the PLC is entered in the HMI tag.
- The HMI tag is configured with the correct data type.

The menu command is not available at a tag with symbolic addressing.

If you select multiple tags, the menu command is available if at least one of the selected tags meets the above-mentioned requirements. Only the tags that meet the requirements are connected.

Requirement

- You have created an HMI tag.
- The tag table is open.
- A PLC tag with the absolute address from the PLC is present.

Procedure

Proceed as follows to reconnect tags:

1. Select the row with the tag in the tag table.
2. Open the shortcut menu and select the menu command "Reconnect PLC tag".
   The system looks for a PLC tag whose absolute address and data type match the settings for the HMI tag. If a matching PLC tag is found, the tag connection is established immediately.

Result

The PLC tag is connected to the HMI tag.

See also

Editing a Tag (Page 2127)

10.2.2.3 Configuring tags

Tag limits

Introduction

You can restrict the value range with limits for numerical tags.

Principle

You can specify a value range defined by a high and low limit for numerical tags.

If the process value violates the value range, you trigger a function list. If the operator enters a value for the tag that is outside the configured value range, the input is rejected. The value is not accepted.

Note

You enter the text of the analog alarms for limit violations in the "Analog alarms" editor.

Application example

Use the limit to warn the operator in good time when the value of a tag enters a critical range, for example.

See also

Applying linear scaling to a tag (Page 2138)

Defining the start value of a tag (Page 2135)

Defining the acquisition cycle for a tag (Page 2142)

Defining Limits for a Tag (Page 2133)

Start value of a tag (Page 2134)

Updating the tag value in Runtime (Page 2136)

Linear scaling of a tag (Page 2137)

Connecting a tag to another PLC (Page 2139)

Indirect addressing of tags (Page 2139)

Addressing tags indirectly (Page 2140)

Using tags to trigger functions (Page 2141)

Address multiplexing (Page 2143)

Configuring address multiplexing with absolute addressing (Page 2145)

Configuring address multiplexing with symbolic addressing (Page 2147)

Creating external tags (Page 2122)

Defining Limits for a Tag

Introduction

For numerical tags, you can specify a value range by defining a low and high limit.

Additionally, you configure the system to process a function list whenever a tag value drops below or exceeds its configured value range.

Requirement

- You created the tag for which you want to set limits.
- The Inspector window with the properties for this tag is open.

Procedure

To define limits of a tag, follow these steps:

1. In the Inspector window select "Properties > Properties > Limits." If you want to define one of the limits as a constant value, select "Constant" using the Ø▾ button. Enter a number in the relevant field.

   If you want to define one of the limits as a tag value, select "HMI tag" using the Ø▾ button. Use the object list to define the tag for the limit.
2. To set an additional limit value for the tag, repeat step 1 with the appropriate settings.

Alternative procedure

You can also configure the high and low limit directly in the tag table. To view hidden columns, activate the column titles using the shortcut menu.

Configuring a function list

You can configure a function list for exceeding the value range as follows:

1. If you want to start a function list when the value drops below the value range, click "Properties > Events > Minimum violated" in the Inspector window. Create a function list in this dialog.
2. If you want to start a function list when the value exceeds the value range, click "Properties > Events > Maximum violated" in the Inspector window. Create a function list in this dialog.

Result

You have set a value range defined by a high and low limit for the selected tag. If the value range is exceeded or undershot, a function list is carried out.

See also

Tag limits (Page 2132)

Start value of a tag

Value of a tag at start of Runtime

You can configure a start value for numeric tags and tags for date/time values. The tag will be preset to this value when Runtime starts. In this way, you can ensure that the tag has a defined status when Runtime starts.

For external tags, the start value will be displayed on the HMI device until it is overwritten by the PLC or by input.

If no start value was configured, the tag contains the value "0" when starting Runtime.

In WinCC Runtime Professional you can enter a tag value in place of the start value on a tag with the "String" data type. The tag value is saved in the "Project texts" editor and is multilingual. After the text has been translated, it is displayed in Runtime as a language-dependent start value.

Application example

You can assign a default value to an I/O field. Enter the desired default value as start value for the tag that is linked to the I/O field.

See also

Defining the start value of a tag (Page 2135)

Tag limits (Page 2132)

Defining the start value of a tag

Introduction

In WinCC you can configure a start value for a numeric tag and a tag for date/time values which this adopts at Runtime start.

Requirement

- You have created the tag for which you want to define a start value.
- The Inspector window with the tag properties is open.

Procedure

To configure a start value, proceed as follows:

1. In the Inspector window select "Properties > Properties > Values."
2. Enter the desired "Start value."

Alternative procedure

You can also configure the start value directly in the tag table. To view hidden columns, activate the column titles using the shortcut menu.

Result

The start value you selected for the tag is transferred to the project.

See also

Tag limits (Page 2132)

Start value of a tag (Page 2134)

Updating the tag value in Runtime

Introduction

Tags contain process values which change while Runtime is running. Value changes are handled differently at internal and external tags.

Principle

When Runtime starts, the value of a tag is equal to its start value. Tag values change in Runtime.

In Runtime, you have the following options for changing the value of a tag:

- A value change in an external tag in the PLC.
- By input, for example, in an I/O field.
- By running a system function, such as "SetValue."

Updating the Value of External Tags

The value of an external tag is updated as follows:

- Cyclic in operation

  If you select the "Cyclic in operation" acquisition mode, the tag is updated in runtime as long as it is displayed in a screen. The acquisition cycle determines the update cycle for tag value updates on the HMI device. Cyclic acquisition is based on the selected scan cycle time.
- Cyclic continuous

  If you select the "Cyclic continuous" acquisition mode, the tag will be updated continuously in Runtime, even if it is not in the currently-open screen. This setting is activated for tags that are configured to trigger a function list when their value changes, for example.

  Only use the "Cyclic continuous" setting for tags that must truly be updated. Frequent read operations increase communication load.
- On demand

  If you select the "On demand" acquisition mode, the tag is not updated cyclically. It will only be updated on request, for example, by using the "Update Tag" system function.

See also

Tag limits (Page 2132)

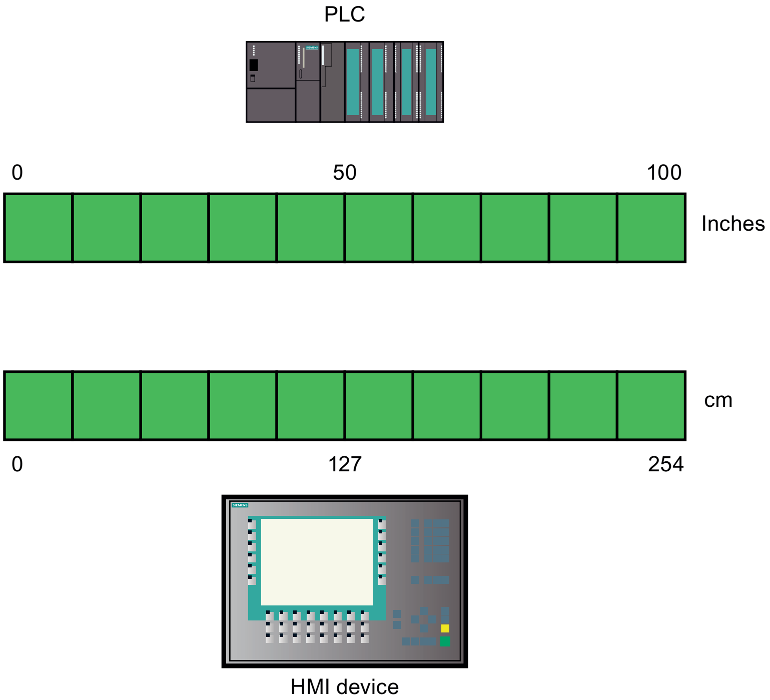
Linear scaling of a tag

Introduction

Numeric data types can be processed with linear scaling. The process values in the PLC for an external tag can be mapped onto a specific value range in the project.

Principle

To apply linear scaling to a tag, you must specify one value range on the HMI device and one on the PLC. The value ranges will be mapped to each other linearly.



As soon as data from the HMI device is written to an external tag, it will be automatically mapped to the value range of the PLC. As soon as data from the HMI device is read from the external tag, a corresponding transformation will be performed in the other direction.

Note

You can also use the system functions "LinearScaling" and "InvertLinearScaling" to automatically convert process values.

Application example

The user enters length dimensions in centimeters but the PLC is expecting inches. The entered values are automatically converted before they are forwarded to the controller. Using linear scaling, the value range [0 to 100] on the PLC can be mapped onto the value range [0 to 254] on the HMI device.

See also

Tag limits (Page 2132)

Applying linear scaling to a tag

Introduction

To apply linear scaling to a tag, you must specify one value range on the HMI device and one on the PLC. The value ranges will be mapped to each other linearly.

Requirement

- The external tag to which linear scaling is to be applied must exist.
- The Inspector window with the properties for this tag is open.

Procedure

To apply linear scaling to a tag, follow these steps:

1. In the Inspector window select "Properties > Properties > Linear scaling."
2. Click on "Enable" to switch on linear scaling.

   Using this option, you can temporarily switch off linear scaling for testing purposes, for example. Settings which were made earlier for linear scaling remain unchanged.
3. In the "PLC" area, enter the start and end values of the value range to be applied to the process values on the PLC.
4. In the "HMI device" area, enter the end and start values of the value range to be applied to the process values on the HMI device.

Result

In Runtime the data will be automatically mapped from one value range to the other.

Note

You can also use the "LinearScaling" and "InvertLinearScaling" system functions to automatically convert process values.

See also

Tag limits (Page 2132)

Connecting a tag to another PLC

Introduction

In WinCC, you can change the PLC connection of a tag at any time. This is needed when you change the configuration of your plant, for example.

Depending on the PLC selected, you may need to modify the configuration of the tag. The tag properties which must be changed will be highlighted in color.

Requirement

- The external tag, whose connection you wish to change, must already exist.
- The connection to the PLC must already exist.
- The Properties window for this tag is open.

Procedure

To change the PLC connection of a tag, proceed as follows:

1. In the Inspector window select "Properties > Properties > General."
2. Select the new connection in the "Connection" field.

   The tag properties that you must change will be highlighted in color in the tag table and in the Inspector window.
3. Change all highlighted properties of the tag to suit the requirements of the new PLC.

Result

The external tag is connected to the new PLC.

See also

Tag limits (Page 2132)

Indirect addressing of tags

Principle

In multiplexes, a type of indirect addressing, the tag used is first determined at runtime. A list of tags is defined for the multiplex tag. The relevant tag is selected from the list of tags in runtime. The selection of the tag depends on the value of the index tag.

In Runtime, the system first reads the value of the index tag. Then the tag which is specified in the corresponding place in the tag list is accessed.

Application example

Using indirect addressing, you could configure the following scenario:

The operator selects one of several machines from a selection list. Depending on the operator's selection, data from the selected machine will be displayed in an output field.

To configure such a scenario, configure the index tag for a symbolic I/O field. Configure the multiplex tag at an I/O field. Configure the tag list of the multiplex tag to reflect the structure of the selection list.

If the operator selects another machine, the value of the index tag will change. The selection field then displays the content of the tag that is pointed to in the tag list in the multiplex tag by the new index value.

See also

Tag limits (Page 2132)

Addressing tags indirectly

Introduction

With indirect addressing, the tag used is first determined at runtime. Instead of a single tag, a list of tags is defined. The list entries consist of an index value and the name of the tag to be used. Using an index tag, you can control which entry in the tag list will be accessed.

Requirement

- The tag which you wish to address indirectly must already exist.
- The index tag must exist.
- The tags which will be contained in the tag list must already exist.
- The Inspector window with the tag properties is open.

Procedure

To address tags indirectly, proceed as follows:

1. In the Inspector window select "Properties > Properties > Multiplexing".
2. Select the "Multiplexing" option to activate indirect addressing.

   Using this option, you can temporarily switch off indirect addressing for testing purposes, for example. Settings which were made earlier for indirect addressing remain unchanged.
3. Select an "Index tag" or define a new tag using the object list.
4. Click the first entry in the "Tags" column in the tag list.
5. Select a tag as a list entry or define a new tag using the object list.

   The entry in the "Index" column will be generated automatically.
6. Repeat step 5 for all tags that you wish to add to the tag list.
7. If necessary, you can use drag-and-drop to change the order of the entries in the list.

Result

In runtime, the system will dynamically access the tag in the tag list which has the same index value as the value currently in the index tag.

See also

Tag limits (Page 2132)

Using tags to trigger functions

Introduction

You can use the values of variables as the triggering event for an action in runtime. To start an action in Runtime, configure a function list for a tag. Include one or more system functions in the function list. The function list is processed when the configured event occurs.

The following events are available for a tag:

- Change in value of the tag

  Function list processing is triggered by each change in the value of the variable.

  If the tag contains arrays, the function list is processed whenever an element of the array changes.
- Violation of the tag's high limit

  The function list is processed when the high limit is violated.
- Violation of the tag's low limit

  The function list is processed when the low limit is violated.

Requirement

- The tag whose value you wish to use as an event already exists.
- The Inspector window with the properties for this tag is open.

Procedure

To configure a tag with a function list, proceed as follows:

1. Under "Properties > Events >" in the Inspector window, select the event for which you want to create a function list.

   The function list associated with the selected event is shown.
2. Click "<Add function>". The second table column contains a selection button.
3. Click the selection button and select a system function.
4. Define the parameter values.

Result

The function list is processed when the configured event occurs in Runtime.

See also

Tag limits (Page 2132)

Defining the acquisition cycle for a tag

Introduction

The value of an external tag can be changed in Runtime by the PLC to which the tag is linked. To ensure that the HMI device is informed of any changes in tag values by the PLC, the values must be updated on the HMI. The value is updated at regular intervals while the tag is displayed in the process screen or is logged. The interval for regular updates is set with the acquisition cycle. The update can also be made continuous.

Requirement

- You have created the tag for which you want to define an acquisition cycle.
- The Inspector window with the tag properties is open.

Procedure

To configure an acquisition cycle for a tag, follow these steps:

1. In the Inspector window select "Properties > Properties > General."
2. If you want to update the tag at regular intervals as long as it is being displayed on the screen or logged, select "Cyclic in operation" as the acquisition mode.
   Or:
   If you want to update the tag at regular intervals even though it is not being displayed on the screen or logged, select "Cyclic continuous" as the acquisition mode.
   The "Cyclic continuous" setting is selected for a tag, for example, that has a function list configured for a change of its value and that is not directly visible in a screen.
3. Select the required cycle time in the "Acquisition cycle" field or define a new acquisition cycle using the object list.

Alternatively, you can configure the acquisition cycle directly in the work area of the tag table. To view hidden columns, activate the column titles using the shortcut menu.

Note

Only use the "Cyclic continuous" acquisition mode for tags that really have to be continuously updated. Frequent read operations generate a heavy communication load.

Result

The configured tag is updated in Runtime with the selected acquisition cycle.

See also

Tag limits (Page 2132)

Address multiplexing

Introduction

Using address multiplexing, you can use a single tag to access a multitude of memory locations within the PLC's address range. You read and write to the addresses without defining a tag for each individual address.

Multiplexing with absolute addressing

When using multiplexing with absolute addressing, you configure tags as placeholders for the address in the PLC to be addressed.

If you want to access, for example, an address of the format "%DBx.DBWy", the expression for multiplexing is as follows:

"%DB[HMITag1].DBW[HMITag2]"

In Runtime, you supply the tag "HMITag1" with the required value for the data block you want to address.

In Runtime, you supply the tag "HMITag2" with the required address from the data block.

Tags are supplied with values, for example, with the help of values from the PLC or via a script.

Multiplexing with absolute addresses is supported for the following PLCs and communication drivers.

- SIMATIC S7 300/400
- SIMATIC S7 1200

Multiplexing with absolute addresses is not available for data blocks with optimized access.

Multiplexing with symbolic addressing

When multiplexing with symbolic addressing, you access an array element of an array tag in a data block of the connected PLC by means of a multiplex tag and an index tag. The multiplex tag contains the symbolic address of the data block which you want to access. The symbolic address also contains the index tag via which you access the index of the array tag.

If you want to access, for example, the array tag "Arraytag_1" in the data block "Datablock_1", the expression for symbolic addressing is as follows:

"Datablock_1.Arraytag_1["HMITag_1"]

You control the access to the index of the array elements with the HMI-Variable "HMITag_1". In Runtime, you supply the tag with the index of the array element that you want to access.

Multiplexing with symbolic addressing is only available if the following components support symbolic addressing:

- the HMI device
- PLC
- Communication driver

Symbolic addressing is currently only supported by the SIMATIC S7 1200 communication driver.

See also

Tag limits (Page 2132)

Configuring address multiplexing with absolute addressing

Introduction

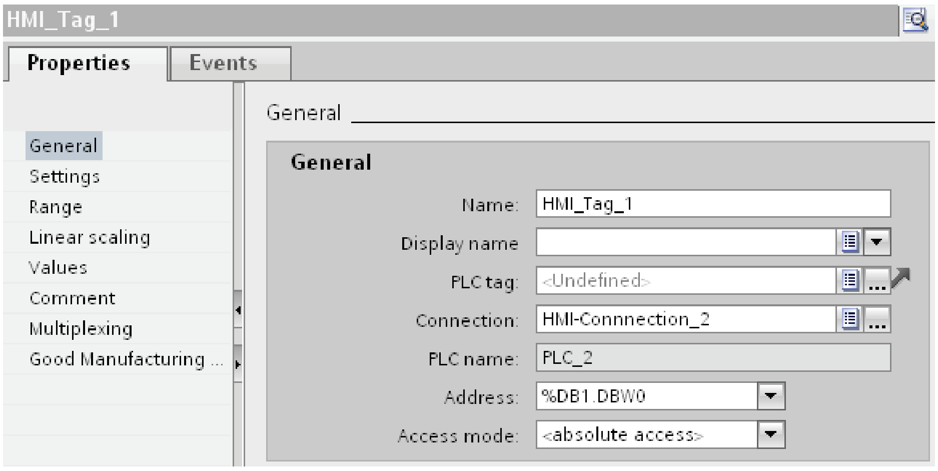
When using address multiplexing, you can efficiently access different addresses in the PLC with the help of a small number of tags. Instead of the absolute address in the PLC, you use tags in order to be able to change the address in Runtime.

Requirement

- The tag for address multiplexing is created and connected to the PLC.
- The Properties window for this tag is open.

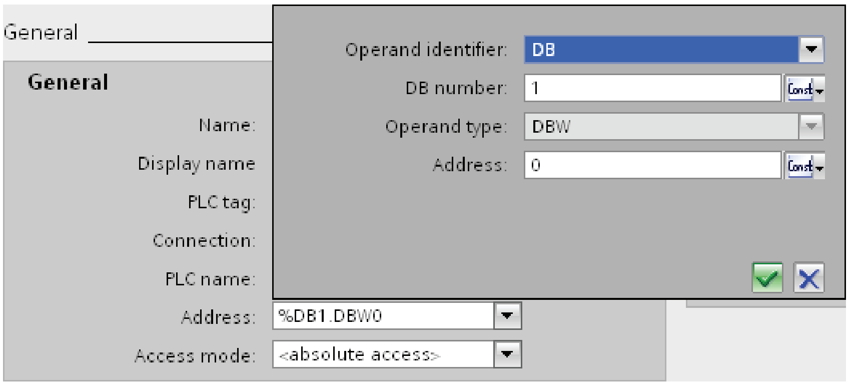
Procedure

1. Select the tag for address multiplexing in the tag table, and select "Properties > Properties > General" in the Inspector window. The general properties of the tag are displayed.



Name: HMI_Tag_1
Display name:
PLC tag: <Undefined>
Connection: HMI-Connnection_2
PLC name: PLC_2
Address: %DB1.DBW0
Access mode: <absolute access>

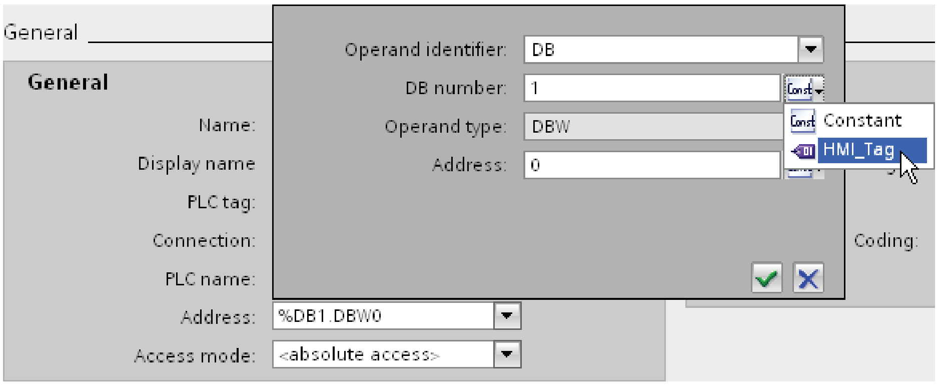
2. Select the "Int" data type for this example.
3. Select the access type "Absolute addressing".
4. Click the selection button in the "Address" field. The address dialog opens.



Operand identifier: DB
DB number: 1
Operand type: DBW
Address: 0

Address: %DB1.DBW0
Access mode: <absolute access>

5. Click the selection button in the "DB number" field and select the entry "HMI tag".



General

General

Name:

Display name

PLC tag:

Connection:

PLC name:

Address: %DB1.DBW0

Access mode: <absolute access>

Coding:

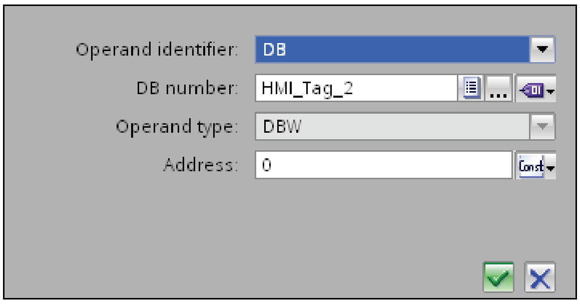
Operand identifier: DB

DB number: 1

Operand type: DBW

Address: 0

6. In the "DB number" field, click the ... button and select a tag for the DB number in the object list. Or create a new tag with the help of the object list. Accept the tag by clicking the ✓ button.



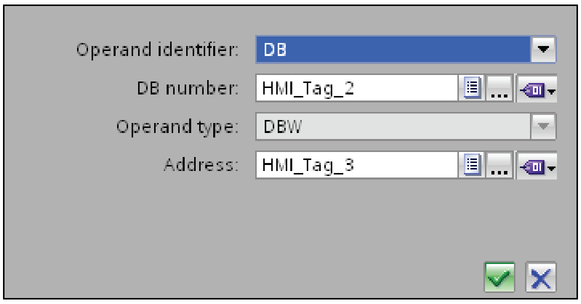
Operand identifier: DB

DB number: HMI_Tag_2

Operand type: DBW

Address: 0

7. Repeat steps 3 and 4 for the "Address" field and configure a further tag for calling the address area in the data block.



Operand identifier: DB

DB number: HMI_Tag_2

Operand type: DBW

Address: HMI_Tag_3

The selection options in the Address dialog depend on the selected data type of the multiplex tag. The Address dialog offers only address settings that can be configured with the selected data type.

Result

In runtime, the multiplex tag is used to access the memory location corresponding to the address currently found in the tag. You control access to the data block with the tag in the DB number field. You control access to the address in the selected data block with the tag in the "Address" field.

Note

The value in the memory location will only be read at the next update cycle for the addressed tag.

If, for example, you use a multiplex tag in a script, do not attempt to access contents of the memory location directly after changing it.

See also

Tag limits (Page 2132)

Configuring address multiplexing with symbolic addressing

Introduction

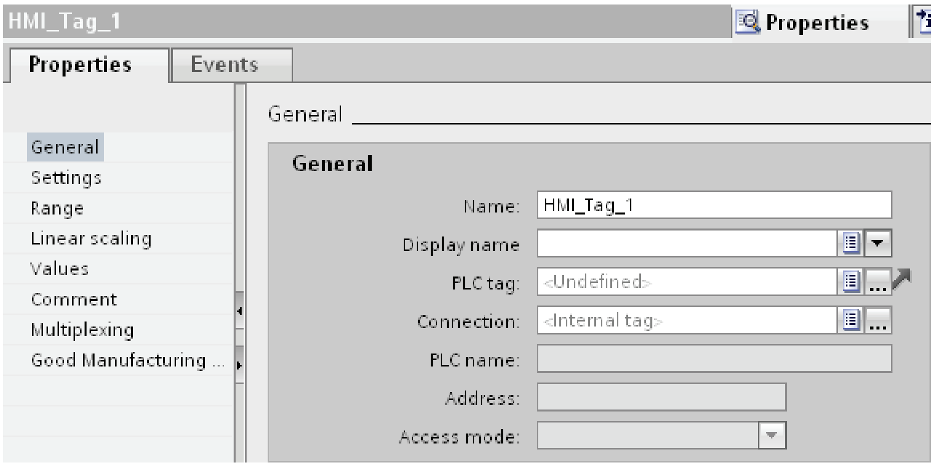
When using address multiplexing, you can efficiently access different addresses in the PLC with the help of a small number of tags. Instead of the symbolic address in the PLC, you use tags in order to be able to change the address in Runtime.

Requirement

- The tag for address multiplexing is created.
- The Properties window for this tag is open.
- A data block with an array tag is created in the connected PLC.
- The data block was compiled.

Procedure

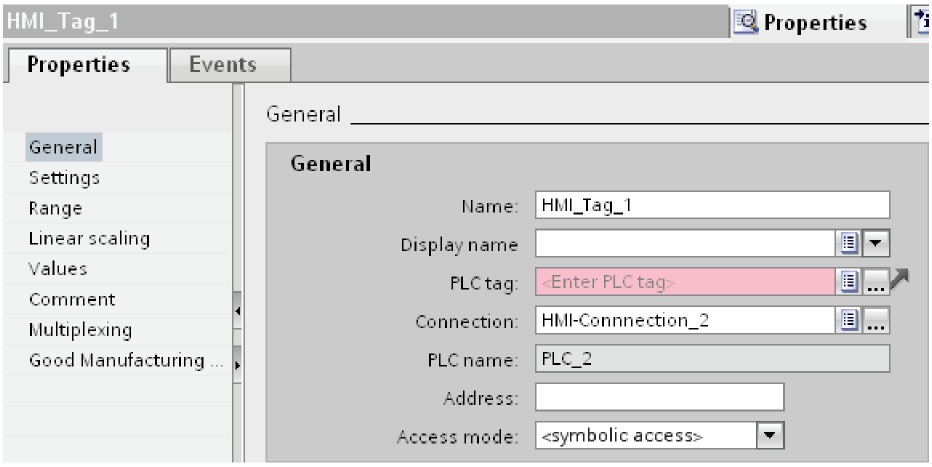
1. Select the tag for address multiplexing in the tag table, and select "Properties > Properties > General" in the Inspector window. The general properties of the tag are displayed.



General

Name: HMI_Tag_1
Display name
PLC tag: <Undefined>
Connection: <Internal tag>
PLC name:
Address:
Access mode:

2. Select the connection to the PLC via the "Connection" field.

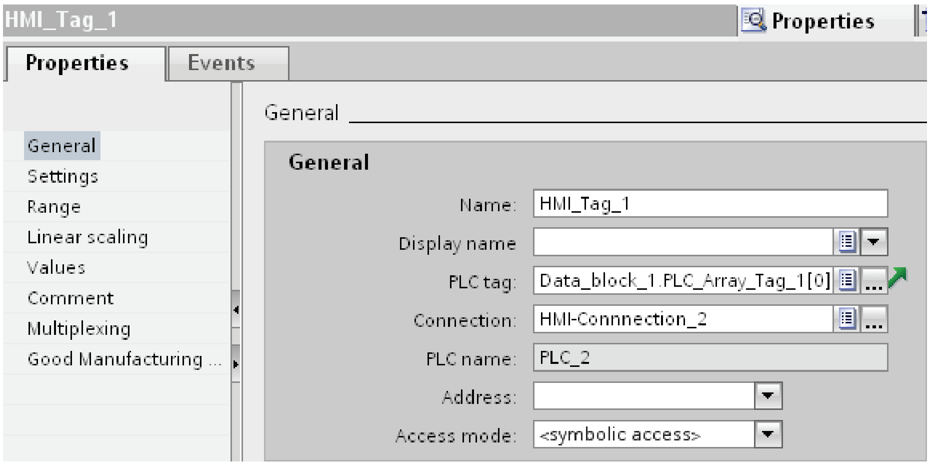


General

Name: HMI_Tag_1
Display name
PLC tag: <Enter PLC tag>
Connection: HMI-Connnection_2
PLC name: PLC_2
Address:
Access mode: <symbolic access>

3. Select the access type "Symbolic addressing".

4. Navigate to the data block of the PLC via the "PLC tag" field and select an array element of the array tag.



General

Name: HMI_Tag_1

Display name:

PLC tag: Data_block_1.PLC_Array_Tag_1[0]

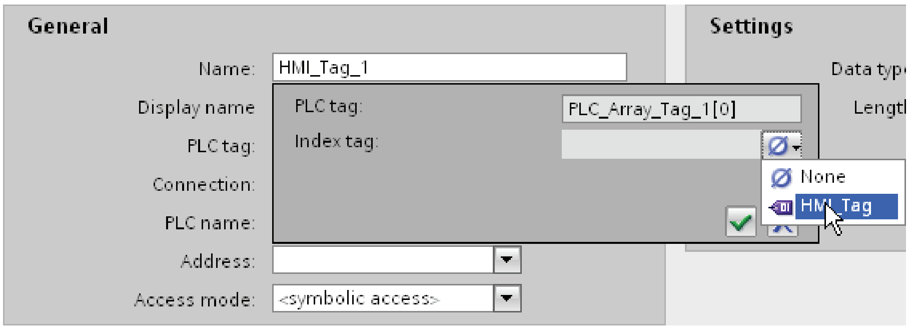
Connection: HMI-Connnection_2

PLC name: PLC_2

Address:

Access mode: <symbolic access>

5. Click the selection button in the "Address" field. The address dialog opens.
6. Click the selection button in the "Index tag" field and select the entry "HMI tag".



Name: HMI_Tag_1

Display name

PLC tag:

Connection:

PLC name:

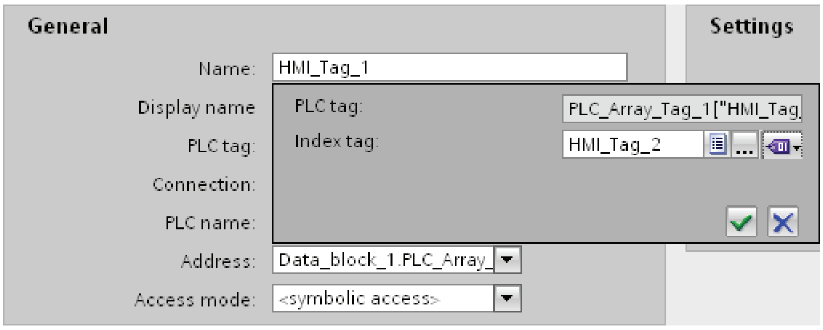
Address:

Access mode: <symbolic access>

PLC tag: PLC_Array_Tag_1[0]

Index tag:

7. In the "Index tag" field, click the ... button and select a tag for the array index in the object list. Or create a new tag with the help of the object list. Accept the tag by clicking the ✓ button.



Name: HMI_Tag_1

Display name

PLC tag:

Connection:

PLC name:

Address: Data_block_1.PLC_Array_

Access mode: <symbolic access>

PLC tag: PLC_Array_Tag_1["HMI_Tag

Index tag: HMI_Tag_2

Result

In Runtime, the array element whose index value is contained in the index tag is accessed.

See also

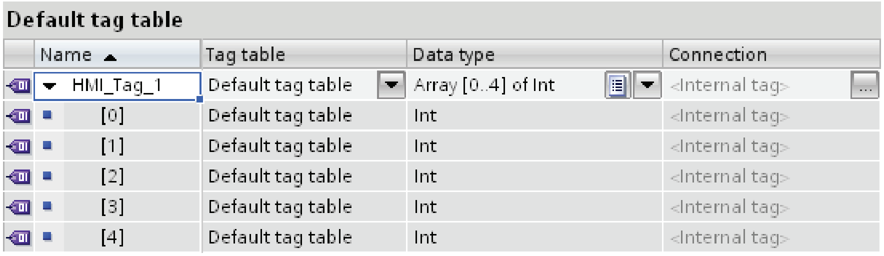
Tag limits (Page 2132)

10.2.3 Working with arrays

10.2.3.1 Basics on arrays

Definition

Array data of a uniform data type is successively arranged and is addressed within the address space to allow access to these data by means of an index. The individual array elements are addressed by means of an integer index. The properties of each array element are the same and are configured at the array tag in a data block of the PLC program.



Default tag table

Name ▲	Tag table	Data type	Connection
▼ HMI_Tag_1	Default tag table	Array [0..4] of Int	<Internal tag>
[0]	Default tag table	Int	<Internal tag>
[1]	Default tag table	Int	<Internal tag>
[2]	Default tag table	Int	<Internal tag>
[3]	Default tag table	Int	<Internal tag>
[4]	Default tag table	Int	<Internal tag>

Advantages

You can configure multiple array elements with the same properties at one time using a single array tag. You can then use each array element as any other tag in your configuration.

Restrictions

The following restrictions apply to the use of arrays:

- Not all HMI devices support array tags.
- An array may contain only one dimension.
- The lower index of an array must begin with "0".

Application examples

Array tags can be used in the following situations:

- To group process values in profile trends: You map process values to trends which are acquired at different points in time, for example.
- To access specific values which are grouped in trends: You output all values of the profile trend by stepping up the index tag, for example.
- To configure discrete alarms with successive bit number.
- To save the complete machine data records in a recipe.

License rule for runtime

An array tag is counted in WinCC Runtime as one PowerTag, regardless of the number of array elements.

Special features

⚠WARNING

Increased system load and performance losses

Read or write access to a single array element always includes read or write access to all array elements of the array tag. Transfer of the data of large arrays from and to the PLC usually takes longer compared to the transfer of a basic data type. This may cause communication overload and disruption as a result.

Example:

- An array tag which consists of 100 array elements of data type "Real" was configured.
- If an array element with a length of four bytes changes, 100 x 4 bytes are written to the PLC.

⚠CAUTION

Data inconsistency at array tags

If the value of a single element must be changed in an array tag, the whole array is read, changed and rewritten as a complete array. Changes carried out in the meantime to other array elements in the PLC are overwritten during rewriting.

You should always prevent the HMI device and the PLC from concurrently writing values to the same array tag. Use synchronous transfer of recipe data records to synchronize an array tag with the PLC.

See also

Creating array tags (Page 2152)

Examples of arrays (Page 2153)

Basics of tags (Page 2115)

10.2.3.2 Creating array tags

Introduction

Create an array tag to configure a large number of tags of the same data type. The array elements are saved to a consecutive address space.

You can create an array tag as an internal tag or as an external tag.

If you want to create an array tag as an external tag, first configure an array tag in a data block of the connected PLC. You then connect the array tag to an HMI tag.

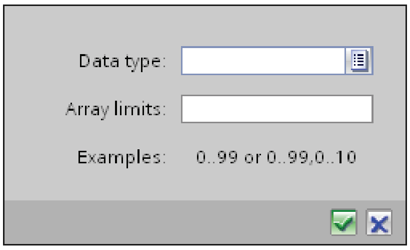
Requirement

- The HMI tag table is open.

Procedure

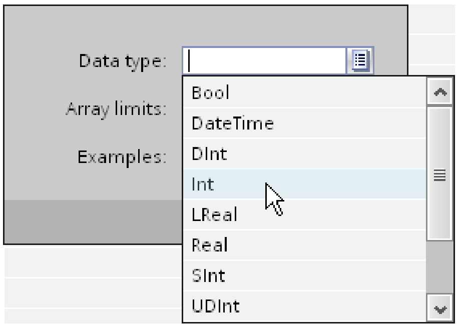
To create an array tag, follow these steps:

1. Double click <Add> in the "Name" column of the HMI tag table.
   A new HMI tag is created.
2. Click ☰ in the Data type column and select the "Array" data type.
3. Click ... in the data type column. The dialog box for configuring the array is opened.

   Data type:
   Array limits:
   Examples: 0..99 or 0..99,0..10

4. Select the desired data type for the array tag in the "Date type" field.

   Data type:
   Array limits:
   Examples:

5. Define the number of array elements in the "Array limits" field. The lower limit must begin with "0".
6. Click ✓. The settings for the array are saved.
7. Save the project.

Result

An array tag is created. The properties of the array elements are inherited from the parent array tag.

See also

Basics on arrays (Page 2150)

10.2.3.3 Examples of arrays

Introduction

Array tags combine a number of tags, e.g. 100 array elements. You use array tags as complete arrays in the following places:

- In the "Alarms" editor
- In the "Recipes" editor
- For address multiplexing
- In the trend view

You use individual array elements everywhere in the configuration like HMI tags.

Examples

You can configure an array tag with the corresponding number of array elements to handle multiple tags of the same data type.

- The individual array elements can be accessed indirectly by means of a multiplex index tag, for example.
- Use these index tags to operate and monitor the array elements.

See also

Basics on arrays (Page 2150)

10.2.4 Working with cycles

10.2.4.1 Cycle basics

Introduction

Cycles are used to control actions that regularly occur in runtime. Common applications are the acquisition cycle and the update cycle.

Principle

In Runtime, actions that are performed regularly are controlled by cycles. Typical applications for cycles:

- Acquisition of external tags
  The acquisition cycle determines when the HMI device will read the process value of an external tag from the PLC. Set the acquisition cycle to suit the rate of change of the process values. The temperature of an oven, for example, changes much more slowly than the speed of an electrical drive.
  Do not set the acquisition cycle too low, since this will unnecessarily increase the communication load of the process.

The smallest possible value for the cycle depends on the HMI device that will be used in your project. For most HMIs, this value is 100 ms. The values of all other cycles are always an integer multiple of the smallest value.

Application example

You can use cycles for the following tasks:

- To regularly update a tag.
- To draw attention to maintenance intervals.

See also

Basics of tags (Page 2115)

10.2.5 Displaying tags

10.2.5.1 Outputting tag values in screens

Introduction

In runtime you can output tag values in the screens of the operator device in the form of a trend. A trend is a graphic representation of the values that a tag takes during runtime. Use the "Trend display" graphic object to represent it. Process values for the trend display are loaded by the PLC from the ongoing process.

The values to be displayed are determined individually within a fixed, configurable cycle. Cyclically-triggered trends are suitable for representing continuous curves, such as the changes in the operating temperature of a motor.

Displayed values

You will need to configure a trend view in a screen so that tag values are displayed on the HMI device. When configuring the trend view, specify which tag values are to be displayed.

You can control the updating of the trend display by defining the cycle time.

10.2.5.2 Configuring trend displays for values from the PLC

Introduction

You use a trend view to graphically represent values that a tag assumes during the process.

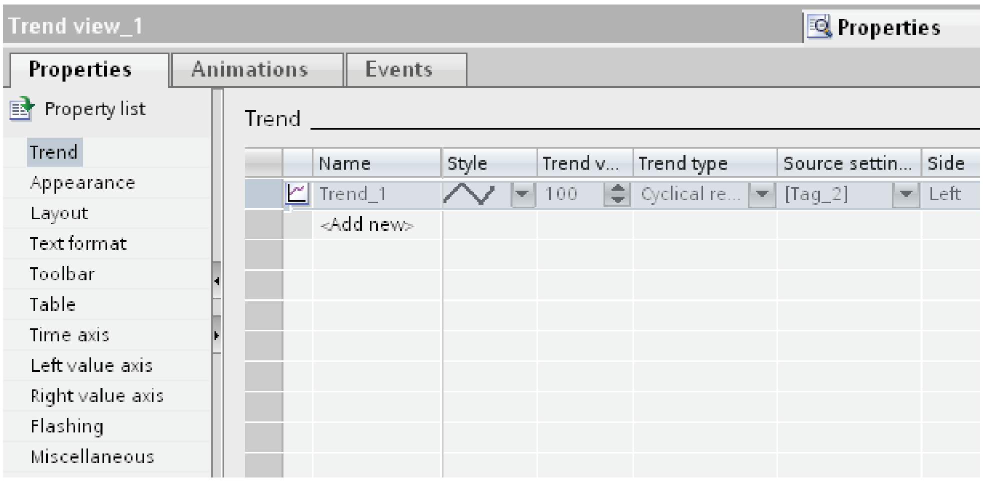
Requirement

- A screen is open.
- The Inspector window with the trend view properties is open.

Procedure

To configure a trend view, follow these steps:

1. Add the "Trend view" object from the toolbox in the "Control" group to the screen.
2. Select the "Trend" category from the "Properties" group in the Inspector window and double-click "<Add>" in the "Name" column.

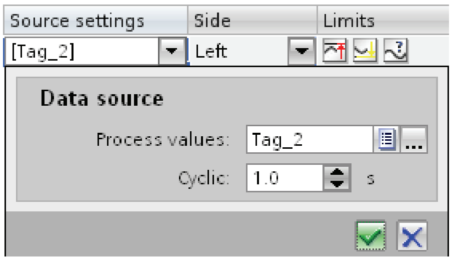


Trend

Name	Style	Trend v...	Trend type	Source settin...	Side
Trend_1		100	Cyclical re...	[Tag_2]	Left
<Add new>					

3. Assign a name to the trend in the "Name" column.
4. In the "Style" column, use the selection button to open the "Style" dialog and select the style of the line.
5. Select the number of trend values in the "Trend values" column.
6. In the "Settings" column, use the selection button to open the "Data source" dialog and select the tags to supply the trend with values.

   Specify the cycle for reading the tags from the PLC.



Source settings	Side	Limits
[Tag_2]	Left	

Data source

Process values: Tag_2

Cyclic: 1.0 s

7. You can make other settings in the dialogs of the Inspector window. For example, you can select the "Display table" option in the "Table" category to display a value table beneath the trend view.

Note

If you hold down the <CTRL> key and double-click the trend view, the trend view is activated. You set the column width and the position of the columns in the table header of the values table in active mode. In order to activate the trend view the zoom factor has to be set to 100 %.

Result

In runtime, the values of the selected tags are displayed in the configured trend view.

10.3 Working with alarms

10.3.1 Basics

10.3.1.1 Alarm system in WinCC

Introduction

The alarm system allows you to display and record operating states and faults on the HMI device that are present or occur in a plant.

An alarm may have the following content:

No.	Time	Date	Alarm text	Status	Alarm class
5	12:50:24:590	24.02.2007	Boiler pressure above high limit.	Incoming Outgoing	Warning: Color Red

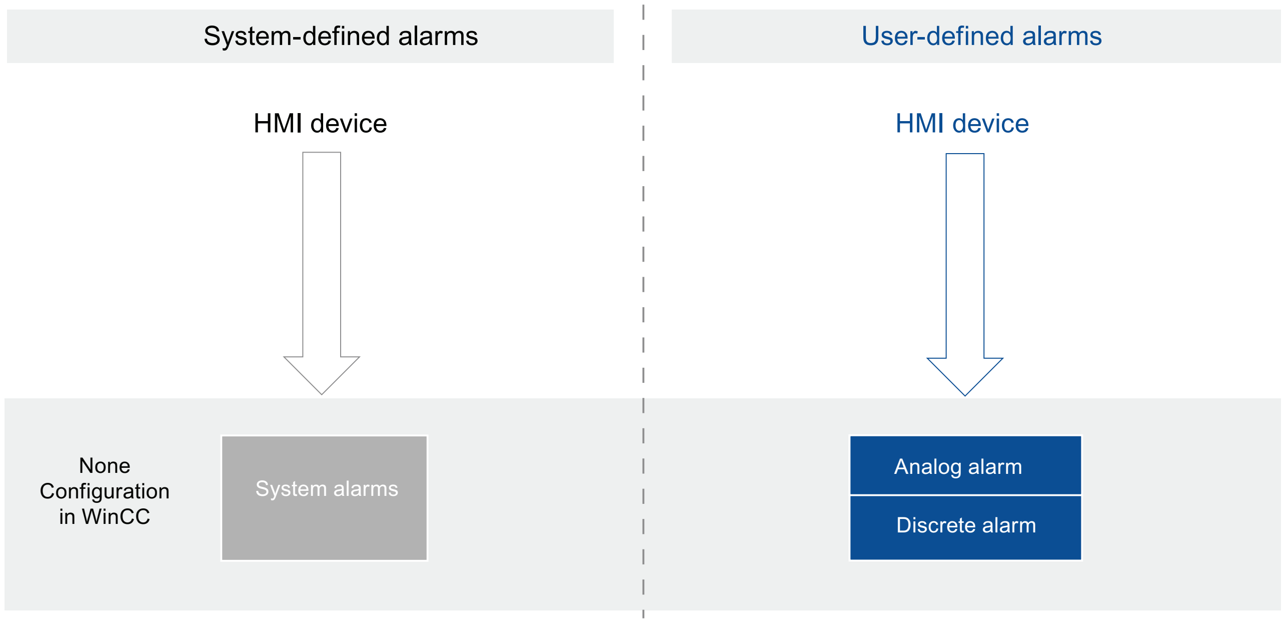
Alarm system in WinCC

The alarm system processes a variety of alarm types. The alarm procedures can be broken down into system-defined alarms and user-defined alarms:

- User-defined alarms are used to monitor the plant process.
- System-defined alarms monitor the HMI device.

The detected alarm events are displayed on the HMI device. Targeted access to the alarms combined with supplementary information about individual alarms ensures that faults are localized and cleared quickly. This reduces stoppages or even prevents them altogether.

The following figure shows the alarm system structure:



10.3.1.2 Alarm types

Overview of the alarm types

Introduction

The alarm types serve various purposes for monitoring the plant. The alarms from the individual alarm types are configured and triggered in different ways.

Select the relevant tab in the "HMI alarms" editor to configure alarms based on the individual alarm types.

Alarm types in WinCC

WinCC supports the following alarm types:

User-defined alarms

- Analog alarms
  - Analog alarms are used to monitor limit violations.
- Discrete alarms
  - Discrete alarms are used to monitor states.

System-defined alarms

- System events
  - System events belong to the HMI device and are imported into the project.
  - System events monitor the HMI device.

System-defined alarms

System alarms

Examples for alarms

- "An online connection to the PLC is established."

Description

A system alarm indicates the status of the system, plus communication errors between the HMI device and system.

Under "Runtime settings > Alarms" you specify how long a system alarm is shown on the HMI device.

Support

The reference contains a list of the possible system events, along with the cause and possible remedies. If you contact online support because of a system alarm on the HMI device, you will need the alarm number and tags used in the system alarm.

User-defined alarms

Analog alarms

Description

Analog alarms signal limit violations during the process.

Example

The speed of the mixer in a fruit juice mixing plant must not be too high or too low. You can configure analog alarms to monitor the speed of the mixer. If the high or low limit for the speed of the mixer is violated, an alarm is output on the HMI device containing the following alarm text, for example: "Mixer speed is too low".

Discrete alarms

Description

Discrete alarms indicate a status in the current process.

Example

A fruit juice mixing plant consists of several tanks containing the ingredients. To ensure the correct mixing ratio of water, fruit concentrate, sugar, and flavoring, the valves in the intakes open and close at the right moment. This operation should be monitored.

You configure a suitable discrete alarm for all the valve states. If a valve on one of the four tanks opens or closes, an alarm is displayed, such as "Water valve closed".

The operator can thus monitor whether the plant is producing correctly.

10.3.1.3 Alarm states

Introduction

An alarm assumes various alarm states in Runtime.

Description

Every alarm has an alarm status. The alarm states are made up of the following events:

- Incoming (I)

  The condition for triggering an alarm is satisfied. The alarm is displayed, such as "Boiler pressure too high."

- Outgoing (O)

  The condition for triggering an alarm is no longer satisfied. The alarm is no longer displayed as the boiler was vented.

- Acknowledge (A)

  The operator acknowledges the alarm.

Alarms without acknowledgment

The following table shows the alarm states for alarms that do not have to be acknowledged:

Display text	Status	Description
I	Incoming	The condition for an alarm is satisfied.
IO	Outgoing	The condition for an alarm is no longer satisfied.

Alarms with acknowledgment

The following table shows the alarm states for alarms that have to be acknowledged:

Display text	Status	Description
I	Incoming	The condition for an alarm is satisfied
IO	Outgoing not acknowledged	The condition for an alarm is no longer satisfied. The operator has not acknowledged the alarm.
IOA	Outgoing and subsequently acknowledged	The condition for an alarm is no longer satisfied. The operator has acknowledged the alarm after this time.
IA	Incoming, acknowledged	The condition for an alarm is satisfied. The operator has acknowledged the alarm.
IAO	Outgoing but acknowledged first	The condition for an alarm is no longer satisfied. The operator acknowledged the alarm while the condition was still satisfied.

Each occurrence of these states can be displayed and logged on the HMI device and a protocol printed.

Note

You can configure the display text for the alarm status.

10.3.1.4 Alarm classes

Basics on alarm classes

Introduction

Many alarms occur in a plant. These are all of different importance. You can assign the alarms of your project to alarm classes to clearly show the user which of the alarms are most important.

Description

The alarm class defines how an alarm is displayed. The alarm class specifies if and how the user has to acknowledge alarms of this alarm class.

A new alarm class with mandatory acknowledgment is generated in WinCC.

Note

The choice of display modes for alarm classes depends on the options on your HMI device.

Examples of how to use alarm classes

- The alarm "Speed of fan 1 in upper tolerance range" has alarm class "Warnings". The alarm is displayed with a white background. The alarm does not have to be acknowledged.
- The alarm "Speed of fan 2 has exceeded upper warning range" is assigned to the "Errors" alarm class. The alarm is displayed with a red background and flashes at high frequency in runtime. The alarm is displayed until the user acknowledges it.

Using alarm classes

Use the following alarm classes to define the acknowledgment model and display of alarms for your project:

- Predefined alarm classes

  You cannot delete predefined alarm classes and edit them only to a limited extent. Predefined alarm classes have been created for each HMI device under "HMI alarms > Alarm classes".
- Custom alarm classes

  You can create new alarm classes under "HMI alarms > Alarm classes", configure how you want the alarms to be displayed, and define an acknowledgment model for alarms of this alarm class. The possible number of custom alarm classes depends on which runtime is used in your project.

See also

Creating alarm classes (Page 2169)

Predefined alarm classes

Predefined alarm classes

The following alarm classes already created in WinCC for every HMI device:

Alarm classes for user-defined alarms

- "Warnings"

  The "Warnings" alarm class is intended to show regular states and routines in the process. Users do not acknowledge alarms from this alarm class.
- "Errors"

  The "Errors" alarm class is intended to show critical or dangerous states or limit violations in the process. The user must acknowledge alarms from this alarm class.

Alarm class for system-defined alarms

- "System"

  The "System" alarm class contains alarms that display states of the HMI device and the PLCs.

See also

Creating alarm classes (Page 2169)

10.3.1.5 Acknowledgment

Acknowledging alarms

Introduction

To make sure that an alarm was registered by the plant operator, configure this alarm so that it is displayed until acknowledged by the operator. Alarms that display critical or hazardous states in the process have to be acknowledged.

Description

Acknowledging an alarm changes the alarm status from "Incoming" to "Acknowledged". When the operator acknowledges an alarm, the operator confirms that he or she has processed the status that triggered the alarm.

Triggering acknowledgment of an alarm

In Runtime, you trigger alarm acknowledgments in various ways:

- Acknowledgement by the authorized user at the HMI device
- Automatic acknowledgment by the system without operator action, e.g. by means of
  - Tags
  - PLC
  - System functions in function lists

Acknowledging alarms that belong together

To make the alarm system clearer and easier to use in Runtime, you can configure an alarm group. You can acknowledge all alarms belonging to this alarm group in a single pass.

Acknowledgment by the PLC

Discrete alarms will be automatically acknowledged by the PLC, if necessary. The acknowledgment is triggered by a bit in the "Acknowledgment tag PLC". You define the bit and tag at the configuration stage.

Acknowledgment of an alarm on the HMI device

In Runtime, the user acknowledges an alarm in one of the following ways, depending on the configuration:

- Using the acknowledgment button <ACK> on the HMI device
- Using the button in the alarm view
- Using configured function keys or buttons in screens

Note

Acknowledgment button <ACK> on the HMI device

To ensure that critical alarms are processed only by authorized users, protect the "ACK" button on the HMI devices, including the operating controls and display objects of the alarms. Use the appropriate operator authorization for this.

Note

HMI device dependency

The acknowledgement key <ACK> is not available on all HMI devices.

Acknowledgment model

Overview

You define the acknowledgment model for an alarm class. Alarms that are assigned to this alarm class will be acknowledged on the basis of this acknowledgment model. The following acknowledgment model is used in WinCC:

- Alarm without acknowledgment

  This alarm comes and goes without having to be acknowledged. There is no visible response from the system.
- Alarm with simple acknowledgment

  This alarm must be acknowledged as soon as the event that triggers the alarm occurs. The alarm remains pending until it is acknowledged.

10.3.1.6 Alarm groups

Introduction

Many alarms from different areas and processes occur in a plant. You can compile associated alarms into alarm groups.

Alarm groups

You can use the alarm groups to monitor the parts of the plant and to acknowledge the associated alarms together as required.

Alarm groups can contain alarms from different alarm classes. You only assign alarms that require acknowledgment to alarm groups.

Using alarm groups

It is a good idea to compile alarm groups for alarms such as the following:

- Alarms that are caused by the same fault.
- Alarms of the same type
- Alarms from a machine unit, such as "Fault in drive XY"
- Alarms from an associated part of the process, such as "Fault in cooling water supply"

Display in Runtime

In Runtime, the "Alarm group" column displays the number of the alarm group to which the alarm belongs.

10.3.1.7 Alarm number

Assigning alarm numbers

The system assigns unique alarm numbers within an alarm type.

Note

When adapting alarm numbers, observe the uniqueness of the alarm number within an alarm type.

10.3.2 Working with alarms

10.3.2.1 Alarm components and properties

Overview

You configure the components of alarms in WinCC. The following table shows the basic components of alarms:

Alarm class	Alarm number	Time of day	Date	Alarm status	Alarm text	Alarm group	Tooltip	Trigger tag	Limit value
Warning	1	11:09:14	06.08.2007	IO	Maximum speed reached	2	This alarm is ...	speed_1	27
System	110001	11:25:58	06.08.2007	I	Switch to "Online" mode	0	This alarm is ...	PLC-Variable_1	–

Alarm class

Alarm classes, such as "Warnings" or "Errors." The alarm class defines the following for an alarm:

- Acknowledgment model
- Appearance in Runtime (e.g. color)

Alarm number

An alarm is identified by a unique alarm number. The alarm number is assigned by the system. You can change the alarm number to a sequential alarm number, if necessary, to identify alarms associated in your project.

Time and date

Every alarm has a time stamp that shows the time and date at which the alarm was triggered.

Alarm status

An alarm has the events "Incoming," "Outgoing," "Acknowledge." For each event, a new alarm is output with the current status of the alarm.

Alarm text

The alarm text describes the cause of the alarm.

The alarm text can contain output fields for current values. The values you can insert depend on the Runtime in use. The value is retained at the time at which the alarm status changes.

Alarm group

The alarm group bundles individual alarms.

Tooltip

You can configure a separate tooltip for each alarm; the user can display this tooltip in Runtime.

Trigger tag

Each alarm is assigned a tag as trigger. The alarm is output when this trigger tag meets the defined condition, e.g. when its state changes or it exceeds a limit.

Limit value

Analog alarms indicate limit violations. Depending on the configuration, WinCC outputs the analog alarm as soon as the trigger tag exceeds or undershoots the limit value.

10.3.2.2 Configuring alarms

Overview of alarm configuration tasks

Steps to configure alarms

Configuring alarms in WinCC involves the following steps:

1. Edit and create alarm classes

   You use the alarm class to define how an alarm will be displayed in runtime and to define the acknowledgment model for it.

2. Creating tags in the "HMI tags" editor

   – Configure the tags for your project.

   – You create range values for the tags.

3. Creating tags in the "HMI alarms " editor

   – Create custom alarms and assign these the tag to be monitored, alarm classes, alarm groups, and other properties.

   – You can also assign system functions or scripts to the alarm events.

4. Output of configured alarms

   To output configured alarms, configure an alarm view or an alarm window in the "Screens" editor.

Additional configuration tasks

Additional tasks may be necessary for configuring alarms, depending on the requirements of your project:

1. Creating alarm groups

   You assign the alarms of your project to alarm groups according to their association, such as by the cause of the problem (power failure) or source of the error (Motor 1).

2. Configuring Loop-In-Alarm

   A Loop-In-Alarm is configured in order to change to a screen containing relevant information on an alarm received.

Creating alarm classes

Introduction

Create alarm classes in the "Alarm classes" tab of the "HMI alarms" editor. Some default alarm classes are already created for every project. You can create additional custom alarm classes. You can create up to 32 alarm classes.

Requirement

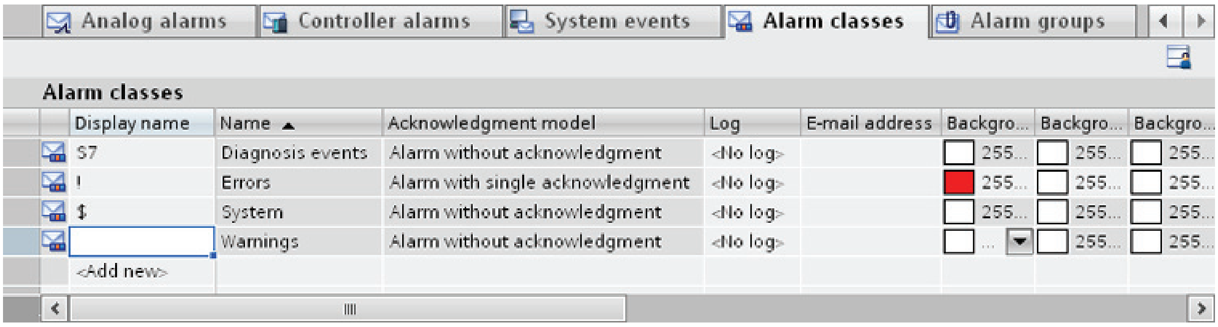
- The "HMI alarms" editor is open.
- The Inspector window is open.

Procedure

To create an alarm class, proceed as follows:

1. Click the "Alarm Classes" tab.

   The predefined and existing custom alarm classes are displayed. The following table lists the predefined alarm classes:



Alarm classes

Display name	Name ▲	Acknowledgment model	Log	E-mail address	Backgro...	Backgro...	Backgro...
S7	Diagnosis events	Alarm without acknowledgment	<No log>		□ 255...	□ 255...	□ 255...
!	Errors	Alarm with single acknowledgment	<No log>		🟥 255...	□ 255...	□ 255...
$	System	Alarm without acknowledgment	<No log>		□ 255...	□ 255...	□ 255...
	Warnings	Alarm without acknowledgment	<No log>		□ ...	□ 255...	□ 255...
<Add new>							

2. Double-click "<Add>" in the table.

   A new alarm class is created. Each new alarm is automatically assigned a static ID.

   The properties of the new alarm class are shown in the Inspector window.
3. Configure the alarm class under "Properties > Properties >General" in the Inspector window.
   – Enter a "Name" and the "Display name".
   – Depending on the HMI device, you can also activate logging, or automatic sending of e-mails.
4. Define the acknowledgment model for the alarm class under "Properties > Properties > Acknowledgment" in the Inspector window.
5. Change the default text under "Properties > Properties > Status" in the Inspector window.

   This text indicates the status of an alarm in Runtime.
6. Change the default colors under "Properties > Properties > Colors" in the Inspector window. Depending on the HMI device, also change the flashing characteristics.

These settings define how alarms from this alarm class are displayed in Runtime.

Note

To display the alarm classes in color in Runtime, the "Use alarm class colors" option must be activated. In the project navigation, enable "Runtime settings > Alarms > General > Use alarm class colors" accordingly. This option is selected in a new project in WinCC.

Configuring alarm groups

Introduction

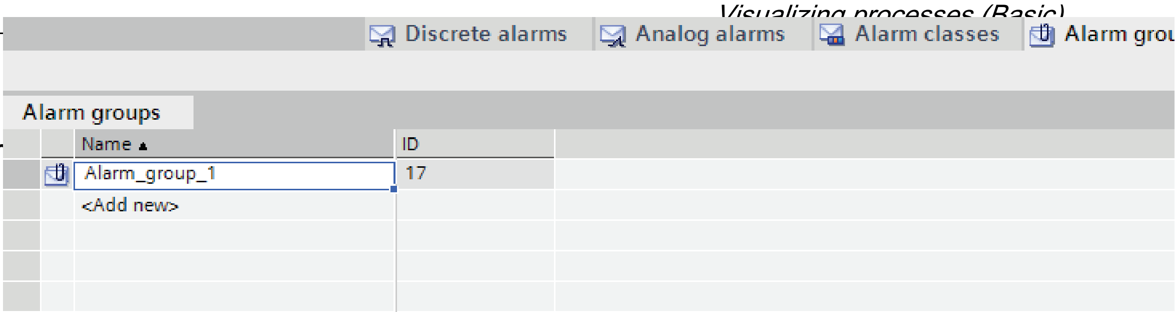
Create alarm groups on the "Alarm Groups" tab in the "HMI alarms" editor. An alarm group is a compilation of single alarms. You assign alarms in an alarm group by association, such as cause of the problem or source of the error. If you acknowledge an alarm from this alarm group in Runtime, all other alarms in the alarm group are acknowledged automatically.

Requirement

- You have created a project.
- The "HMI alarms" editor is open.
- The Inspector window is open.

Creating a new alarr

1.



Alarm groups

Name ▲	ID
Alarm_group_1	17
<Add new>	

2. In the work area of the table, double-click "<Add>" in the first free row.

   A new alarm group is created.

3. You can overwrite the proposed "Name".

Result

An alarm group is created. For the group acknowledgment of alarms in Runtime, assign the associated alarms that require acknowledgment to an alarm group.

Configuring discrete alarms

Introduction

Discrete alarms triggered by the PLC indicate status changes in a plant. They indicate the opened or closed state of a valve, for example.

The following sections describes the configuration procedures in the "HMI alarms" editor. You can also configure discrete alarms in the "HMI tags" editor.

Requirements

- The "HMI alarms" editor is open.
- The Inspector window is open.
- You have created the required alarm classes and alarm groups.

Procedure

To configure a discrete alarm, proceed as follows:

1. Open the "Discrete alarms" tab.
2. To create a new discrete alarm, double-click in the work area on "<Add>".

   A new discrete alarm is created.
3. To configure the alarm, select "Properties > Properties >General" in the Inspector window:
   - Enter an alarm text as event text.

     Use the functions of the shortcut menu to format the text on a character-by-character basis, or to insert output fields for HMI tags, or texts from the text lists.
   - You can renumber the alarm.
   - Select the alarm class and the alarm group, if necessary.
4. In the Inspector window, select the tag and the bit that triggers the alarm under "Properties > Properties > Trigger". Note the following information:
   - Use the data types "Int" or "UInt" to select an HMI tag.
   - Use the data types "Int" or "Word" to select a PLC tag.
   - Use trigger tag bits only for alarms.
   - Do not use trigger tags for anything else.
   - If you want to acknowledge the alarm via the PLC, use this tag also as PLC acknowledgment tag.

NOTICE
Note the method used to count bits in the utilized PLC when specifying the bit. For more information, refer to the "Communication" section in the PLC Online Help.

Note

If the object does not yet exist in the selection list, create it directly in the object list and change its properties later.

Status-dependent alarm texts

To display a different text independent of the alarm status, link a text list to the alarm text. You control the text list with a tag.

Additional settings for discrete alarms

Creating a tooltip

To configure a tooltip for the alarm, follow these steps:

- Enter your text under "Properties > Properties > Tooltip".

Configuring event-driven tasks

To configure event-driven tasks, such as a loop-in alarm, follow these steps:

1. Select the discrete alarm.
2. Select "Properties > Events" in the Inspector window and configure a new function list for the relevant event.

See also

Configuring loop-in alarm (Page 2178)

Configuring analog alarms

Introduction

Analog alarms indicate limit violations. For example, if the speed of a motor drops below a certain value, an analog alarm is triggered.

Requirements

- The "HMI alarms" editor is open.
- The Inspector window is open.
- You have created the required alarm classes and alarm groups.

Procedure

To configure an analog alarm, proceed as follows:

1. Click the "Analog alarms" tab.
2. To create a new analog alarm, double-click in the table on "<Add>".

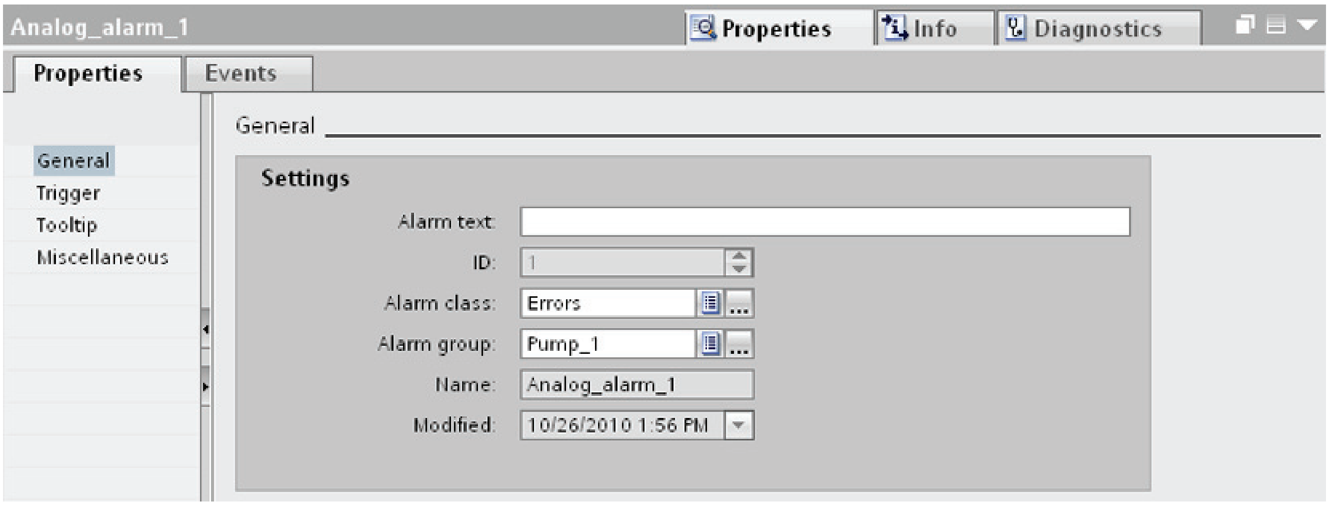
   A new analog alarm is created.
3. To configure the alarm, select "Properties > Properties >General" in the Inspector window:
   – Enter an alarm text as event text.

     Format the text character-for-character using the shortcut menu.

     Using the shortcut menu, you can insert output fields for HMI tags, or text from text lists.

   – You can renumber the alarm.

   – Select the alarm class and the alarm group, if necessary.



Settings

Alarm text:

ID: 1

Alarm class: Errors

Alarm group: Pump_1

Name: Analog_alarm_1

Modified: 10/26/2010 1:56 PM

4. Configure the tag that triggers the alarm under "Properties > Properties > Trigger > Settings".

   Do not use trigger tags for anything else.

Configure limit values for an analog alarm

1. In the Inspector window, click the button under "Properties > Properties > Trigger > Limit > Value".
   – To use a constant as limit value, select "Constant".

     Enter the required limit value.

   – To use a tag as limit value, select "HMI tag".

     The button is shown. Use this button to select the tag you want to use.

Note

If the tag included in the selection does not yet exist, create it in the object list and change its properties later.

2. Select the mode:
   - "High limit violation": The alarm is triggered when the limit is exceeded.
   - "Low limit violation": The alarm is triggered when the limit is undershot.

Optional settings for analog alarms

Setting the delay time

To set the delay time, proceed as follows:

- Enter a time period in the Inspector window under "Properties > Properties> Trigger > Settings > Delay".

  The alarm is only triggered when the trigger condition is still present after the delay time has elapsed.

Setting the deadband

Note

If a process value fluctuates around the limit, the alarm associated with this fault may be triggered multiple times. To prevent this from happening, configure a deadband or delay time.

To enter the deadband, follow these steps:

1. Under "Properties > Properties> Trigger > Deadband > Mode", select the change in alarm status for which the deadband is to be taken into account.
2. Enter a constant value under "Value".
3. To define the deadband value as a percentage of the limit, set the "in %" check box.

Creating a tooltip

To configure a tooltip for the alarm, follow these steps:

- Select "Properties > Properties > Tooltip" in the Inspector window and enter your text.

Configuring event-driven tasks

To configure event-driven tasks, such as a loop-in alarm, follow these steps:

1. Select the analog alarm.
2. Select "Properties > Events" in the Inspector window and configure a new function list for the relevant event.

See also

Configuring loop-in alarm (Page 2178)

Adding an output field to alarm text

Introduction

In WinCC, you can insert output fields into the alarm text which display the content of tags.

Requirements

- The "HMI alarms" editor is open.
- The alarm is selected.

Output of a tag value in the alarm text

To insert an output field for a tag value in the alarm text, proceed as follows:

1. Place the cursor onto the required position in the event text.
2. Select "Insert tag output field" in the shortcut menu.
3. Open the object list under "Tag" and select a tag.

   You can also create the tag in the object list.
4. Under "Format", specify the length of the output field and the format for tag value output in the alarm text.

   Configure an output field of sufficient size. Otherwise, the tag content is not output to the full extent in the alarm.
5. Click ✓ to save your entries.

WinCC inserts a placeholder for the output field into the alarm text: "<tag: n, [tag name]>" whereby n = text string length.

Editing output field properties

To edit the properties of an output field, proceed as follows:

- Double-click on the output field in the alarm text and edit the settings.

Deleting an output field from the alarm text

To delete an output field from the alarm text, proceed as follows:

- Select the output field in the alarm text and then select the "Delete" command from the shortcut menu.

Note

The sequence of the tag output fields in the alarm text depends on the language. Changing the tag of an output field in one language causes the modified output field to appear at the end of the alarm text in all other languages.

Formatting alarm text

Requirements

- The "HMI alarms" editor is open.
- An alarm has been created.

Procedure

To format an alarm text, proceed as follows:

1. Select the alarm to edit.
2. In the Inspector window, select the characters to format under "Properties > Properties > General > Alarm text".
3. Select the formatting from the shortcut menu, e.g. "Underscored" or "Uppercase".

Result

The selected characters are displayed in Runtime with the selected formatting.

Removing format settings

To remove all text formats, proceed as follows:

1. In the Inspector window, select the characters whose formatting you want to remove in the alarm text.
2. Select "Delete formatting characters" from the shortcut menu.

Result

The selected characters are displayed in unformatted notation in Runtime.

Configuring loop-in alarm

Introduction

A Loop-In-Alarm is configured in order to change to a screen containing relevant information on an alarm received.

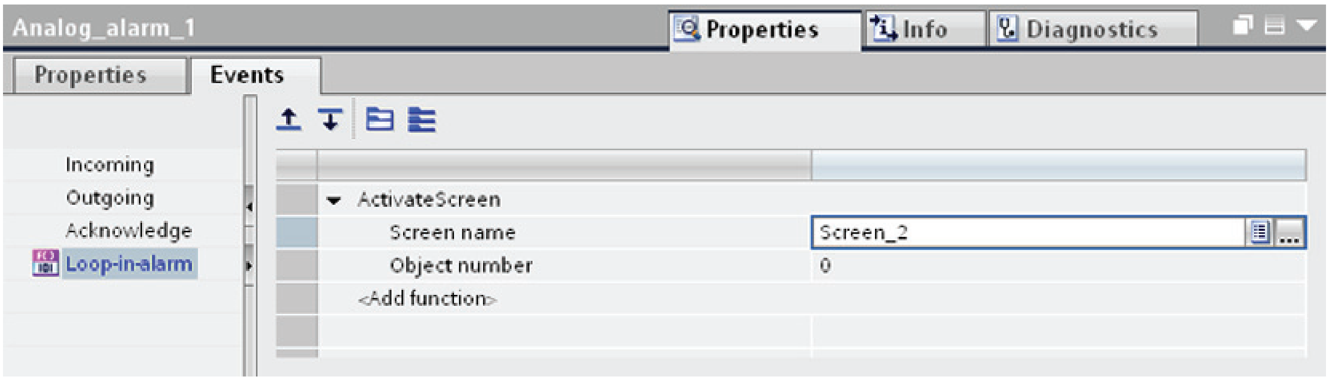
Requirements

- The screen called by the Loop-In-Alarm has been created.
- The "HMI alarms" editor is open.

Procedure

To configure a Loop-In-Alarm for an alarm, proceed as follows:

1. Click the tab that contains the alarm for which you want to configure the Loop-In-Alarm.
2. Select the alarm.
3. In the Inspector window, select "Properties > Events > Loop-In-Alarm".
4. Select the "ActivateScreen" system function.
5. Select the screen called by the Loop-In-Alarm as parameter.



ActivateScreen
Screen name: Screen_2
Object number: 0
<Add function>

Note

To configure the Loop-In-Alarm for an alarm view with an "alarm line" format, use the following system functions:

- "EditAlarm" for HMI devices with keys
- "AlarmViewEditAlarm" for HMI devices without keys

The system functions trigger the "Loop-In-Alarm" event. The alarm line has no buttons.

Result

If you click on the "Loop-In-Alarm" button of the alarm view in Runtime, a screen is opened with information on the selected alarm.

See also

Configuring analog alarms (Page 2173)

Configuring discrete alarms (Page 2171)

Alarms in the "HMI tags" Editor

Configuring discrete alarms in the "HMI tags" editor

Introduction

In WinCC, you can create and edit discrete and analog alarms, including the trigger tags, in the "HMI tags" editor.

Note

If you delete, move or copy objects in the "HMI tags" editor, the changes also take effect in the "HMI alarms" editor.

Requirements

The "HMI tags" editor is open.

Procedure

To configure a discrete alarm, proceed as follows:

1. To create a tag, click on "<Add>" in the table at the top of the work area.

   A new tag is created.
2. Configure an internal or external tag as required.
   – Use the data types "Int" or "UInt" to select an HMI tag.
   – Use the data types "Int" or "Word" to select a PLC tag.
3. Select the tag at the top of the work area.
4. Click on "<Add>" in the table on the "Discrete alarms" tab at the bottom of the work area.

   A new discrete alarm is created for the tag. If you have selected the incorrect data type, the tag will be marked in the discrete alarm.
5. Configure the discrete alarm in the Inspector window:
   – Enter the alarm text under "Properties > Properties > General > Alarm text".

     You can also insert output fields into the alarm text.
   – Select an alarm class.
   – Select the trigger bit of the tag that triggers the discrete alarm under "Properties > Trigger".

6. You can create additional discrete alarms to monitor the tags.

> **Note**
>
> A tag is monitored using only one alarm type. You should therefore create either analog alarms **or** discrete alarms for a tag.

Result

The configured discrete alarms are created in the "HMI tags" editor and displayed in the "HMI alarms" and "HMI tags" editors.

Configuring analog alarms in the "HMI tags" editor

Introduction

In WinCC, create the discrete and analog alarms, including the trigger tags, in the "HMI tags" editor. You can also edit the alarms as in the "HMI alarms" editor. You can create up to two range values for a tag. You monitor these limits with analog alarms.

Requirements

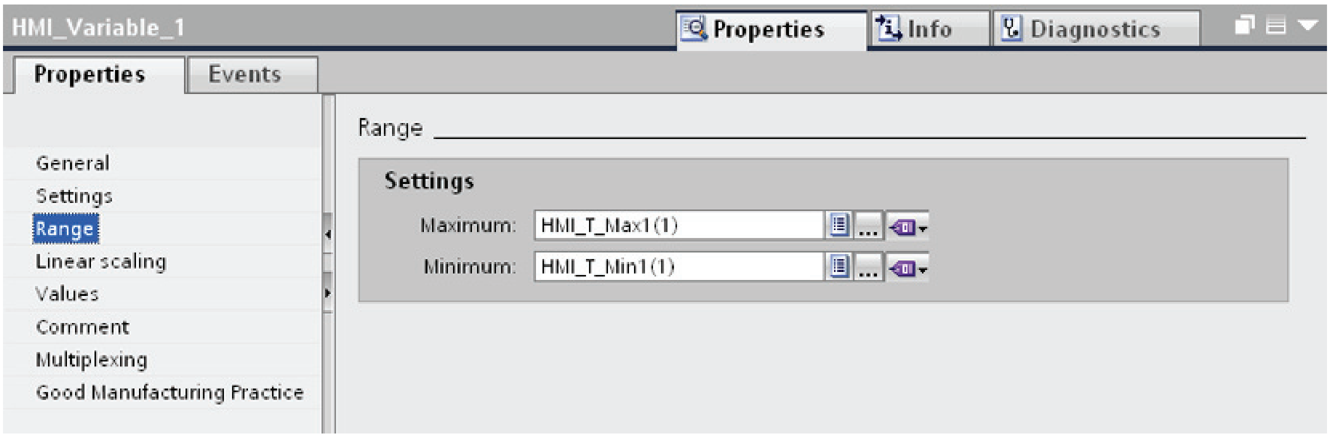
The "HMI tags" editor is open.

Procedure

To configure an analog alarm in the "HMI tags" editor, proceed as follows:

1. To create a tag, click on "<Add>" in the table at the top of the work area.

   A new tag is created.
2. Configure an internal or external tag as required.
3. In the Inspector window, configure the range values of the tags under "Properties > Properties > Range":
   - Select whether to use a "Constant" or an "HMI tag" as limit value for your range values. The object list opens when you select "HMI tag". Select the tag you want to use.



Settings

Maximum: HMI_T_Max1(1)

Minimum: HMI_T_Min1(1)

4. Click the "Analog Alarms" tab at the bottom of the work area.

   Create an analog alarm for both range values.
5. Select an analog alarm and configure it in the Inspector window:
   – Enter the alarm text under "Properties > Properties > General > Alarm text".
   – You can also insert output fields into the alarm text.
   – You can change the default alarm class.
6. Configure the analog alarms as in the "HMI alarms" editor.
7. Complete the configuration of all analog alarms.

Note

A tag is monitored using only one alarm type. You should therefore create either analog alarms **or** discrete alarms for a tag.

Result

The configured analog alarms are created in the "HMI tags" editor and displayed in the "HMI alarms" and "HMI tags" editors.

10.3.2.3 Configuring alarm output

Overview of configuring alarm output

Steps to complete when configuring alarm output

You configure the alarm output in WinCC in the following steps:

1. Create alarm view

   Use the display and control objects in the "Screens" editor to display alarms in Runtime.

2. Configure acknowledgment

   In the "Screens" editor, you can set the operator action that will trigger the acknowledgment.

Additional configuration tasks

Additional tasks may be necessary for configuring alarm views, depending on the requirements of your project:

1. Setting up authorizations

   To make sure only authorized operators process the alarms, assign authorizations for the alarm view and the function keys of the HMI device.

2. Configuring the filtering of the alarm view

   You configure the filtering of the alarms in Runtime in the "Screens" editor. You can also configure alarm views that only display selected alarms.

3. Configure operator input alarms

   Configure the operator input alarms on the operator controls of the HMI device in the "Screens" editor. A preconfigured operator input alarm is output for an operator action. An operator input is, e.g. the acknowledgment of an alarm.

Displaying alarms

Options for displaying alarms on the HMI device

WinCC offers the following options for displaying alarms on the HMI device:

- Alarm view

  The alarm view is configured in a screen. More than one alarm can be displayed simultaneously, depending on the configured size. You can configure multiple alarm views with different contents.

- Alarm window

  The Alarm window is configured in the "Global screen" editor. The alarm window can display multiple alarms at the same time, depending on the configured size. An event can trigger closing and reopening of the alarm window. To hide it during configuration, create an alarm window on its own level.

Additional signals

- Alarm indicator

  The alarm indicator is a configurable, graphical icon. When an alarm comes in, the alarm indicator is displayed on the HMI device. You configure the alarm indicator in the "Global screen" editor.

  The alarm indicator has two states:

  – Flashing: At least one alarm that requires acknowledgment is pending.

  – Static: The alarms are acknowledged but at least one of them has not gone out yet.

    The alarm indicator also displays the number of pending alarms according to the HMI device.

- System functions

  You can configure a list of functions for the event associated with an alarm. These functions must be executed in Runtime when the event occurs.

  Use system functions for alarms in WinCC to control the alarm view or the alarm window other than via the toolbar.

Displaying the predefined alarm classes in Runtime

The following table shows the symbols used to display the predefined alarm classes in the alarm view:

Alarm class	Displayed icon
"Errors"	!
"System"	$
"Warnings"	<No symbol>

Configuring an alarm view

Introduction

Current alarms are displayed in Runtime in an alarm view or alarm window.

Requirement

- A screen is open in the "Screen" editor.
- The "Tools" task card is open.

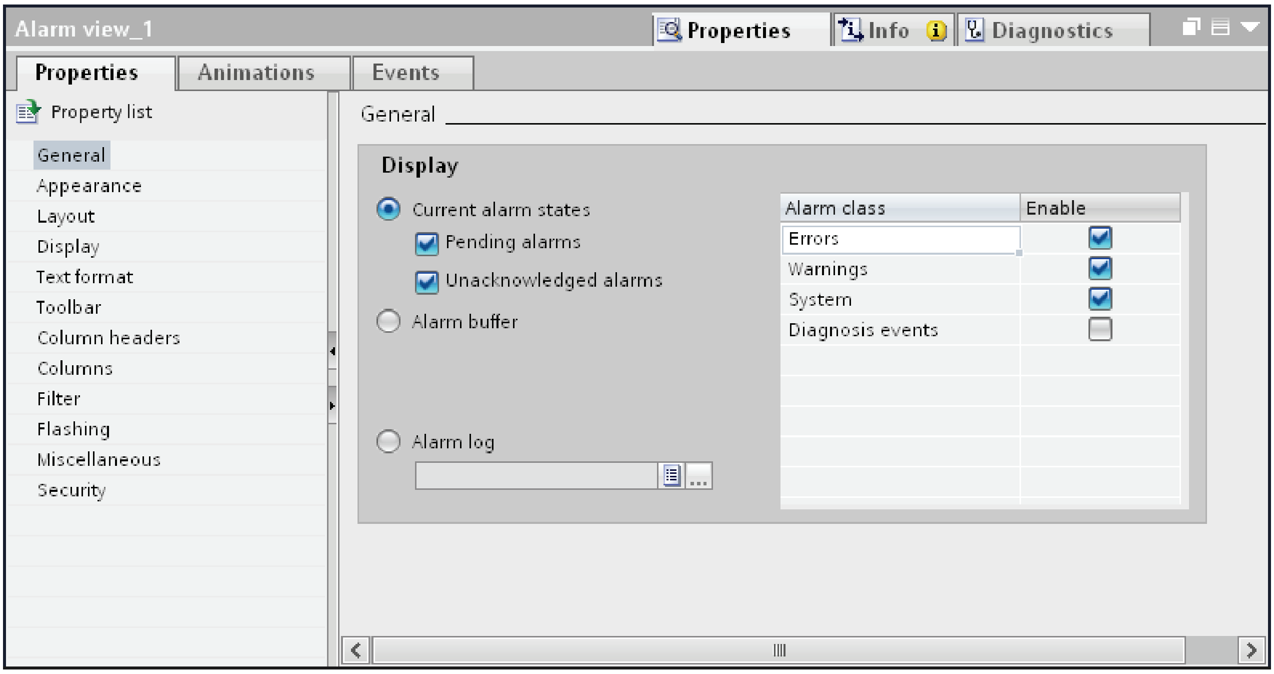
Configuring alarms for the alarm view

To specify the alarms that will be shown in the alarm view, proceed as follows:

1. Insert an "Alarm view" object from the "Tools" task card into the screen.
2. Select the alarm view.
   – In the Inspector window, select "Properties > Properties > General > View > Current alarm states".

     Set whether to display alarms with and/or without mandatory acknowledgment.

   – To display all alarms in the alarm buffer, enable "Alarm buffer".



● Current alarm states
☑ Pending alarms
☑ Unacknowledged alarms
○ Alarm buffer
○ Alarm log

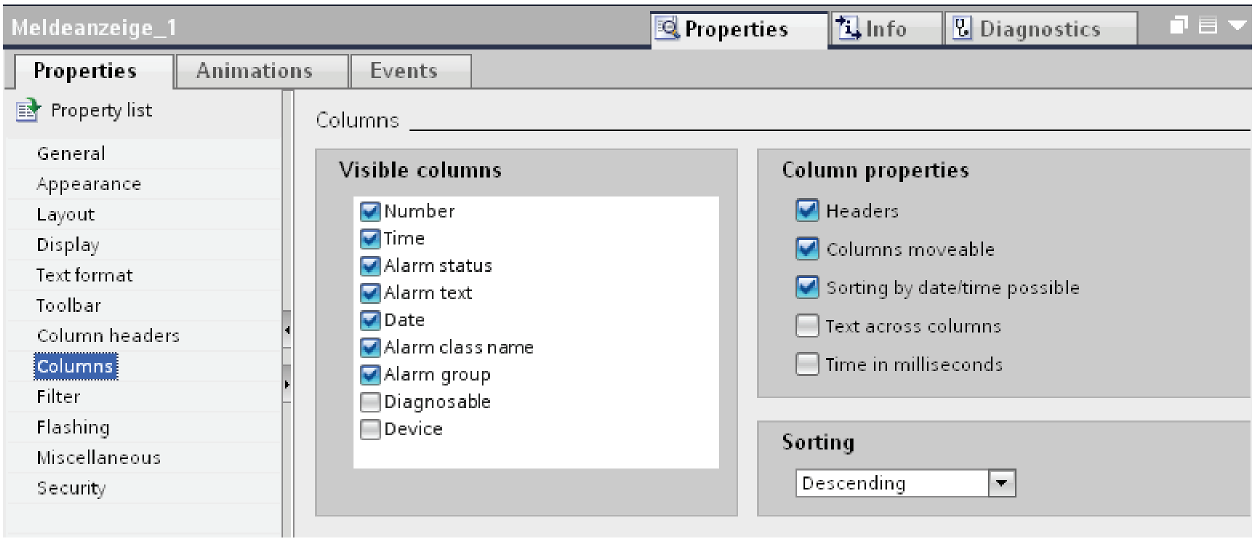
Alarm class	Enable
Errors	☑
Warnings	☑
System	☑
Diagnosis events	☐

3. In the table, activate the alarm classes to be displayed in the alarm view.

Configuring the layout of the alarm view

To specify how the alarms are shown in the alarm view, proceed as follows.

1. Under "Properties > Properties > Layout > Settings > Lines per alarm" in the Inspection window, specify the number of lines to display for each alarm.
2. In "Properties > Properties > View", select the control elements that are available on the HMI device.
3. Configure the columns under "Properties > Properties > Columns":
   – Under "Visible columns" select the columns to be output in the alarm view.
   – Under "Properties Column", define the properties of the columns.
   – Under "Sort", select the sorting order of the alarms.



Columns

Visible columns

- ☑ Number
- ☑ Time
- ☑ Alarm status
- ☑ Alarm text
- ☑ Date
- ☑ Alarm class name
- ☑ Alarm group
- □ Diagnosable
- □ Device

Column properties

- ☑ Headers
- ☑ Columns moveable
- ☑ Sorting by date/time possible
- □ Text across columns
- □ Time in milliseconds

Sorting

Descending

Result

Alarms of various alarm classes are output in the alarm view during runtime.

Configuring an alarm window

Introduction

The alarm window displays current alarms. The alarm window is configured in the "Global Screen" editor and opens regardless of the current screen. The HMI device can still be used, even if alarms are pending and displayed. An alarm window is displayed and configured like an alarm view.

To hide an alarm window during configuration, create it on its own level.

Requirement

- The "Global Screen" editor is open.
- The "Tools" task card is displayed.
- The Inspector window is open.

Procedure

Proceed as follows to configure an alarm window:

1. Insert an "Alarm window" object from the "Tools" task card into the global screen.
2. Configure the alarm window like an alarm view.
3. Under "Properties > Properties > Mode > Window" in the Inspector window, select how the alarm window reacts and is operated in Runtime.
   – Activate "Modal" if the alarm window is to retain the focus in Runtime after a screen change.

     This option is important, as switching back and forth between the screen and different windows with <Ctrl+TAB> is not supported.

Result

During runtime, the alarms of the selected alarm class are displayed in the alarm window.

Configuring an alarm indicator

Introduction

The alarm indicator uses a warning triangle to indicate that alarms are pending or require acknowledgement. If an alarm of the configured alarm class occurs, the alarm indicator is displayed.

The alarm indicator has two states:

- Flashing: At least one alarm that requires acknowledgment is pending.
- Static: At least one of the acknowledged alarms has not gone out yet.

During configuration, specify whether Runtime has to open an alarm window when you operate the alarm indicator.

Requirement

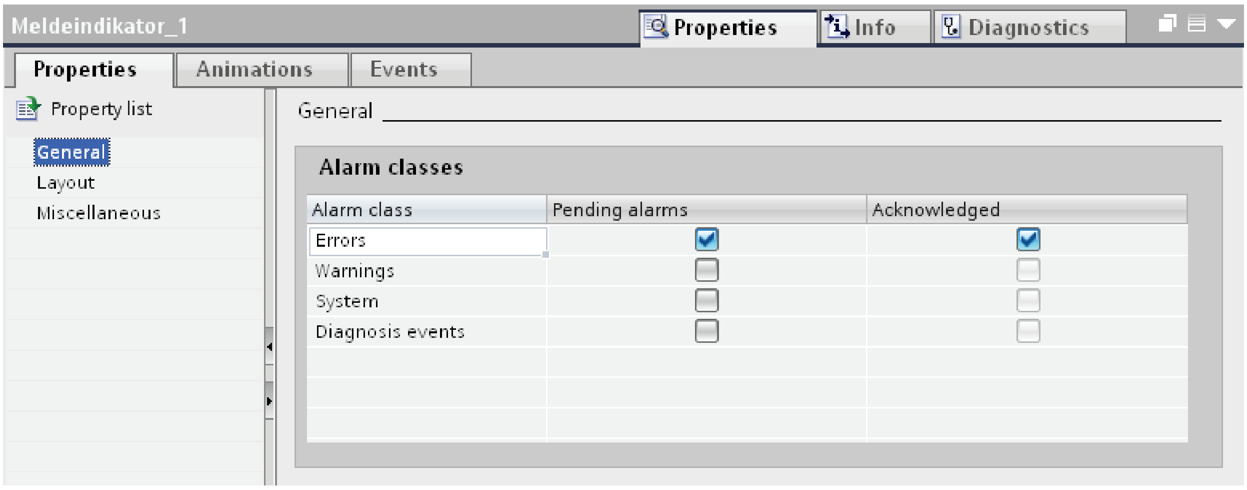
- The "Global Screen" editor is open.
- The "Tools" task card is open.
- The Inspector window is open.

Procedure

Proceed as follows to configure the alarm indicator:

1. Insert the "Alarm indicator" object from the "Tools" task card into the work area.
2. Select the alarm indicator.
3. Under "Properties > Properties > General" in the Inspector window, select the alarm classes to be displayed by the alarm indicator.

Specify whether to display pending and/or acknowledged alarms in the alarm indicator.



Alarm classes

Alarm class	Pending alarms	Acknowledged
Errors	☑	☑
Warnings	□	□
System	□	□
Diagnosis events	□	□

4. Under "Properties > Event", assign system function "ShowAlarmWindow" to an event of the alarm indicator.

Note

If you have configured a permanent window in the screen or template, do not position the alarm window and alarm indicator in the vicinity of the permanent window. Otherwise the alarm window and the alarm indicator are not displayed in Runtime. However, the permanent window is not visible in the "Global screen" editor.

Result

The alarm indicator is displayed if alarms from the selected alarm class are pending or need to be acknowledged in Runtime. The alarm window opens when the user operates the alarm indicator.

10.3.2.4 Acknowledging alarms

Configuring alarm acknowledgment by means of alarm class

Introduction

To configure an alarm with alarm acknowledgment, assign it to an alarm class with the "Alarm with single acknowledgment" acknowledgment model.

Requirement

- The "HMI alarms" editor is open.
- The required alarm class has been created.
- The required alarm has been created.

Selecting the acknowledgment model for an alarm class

The acknowledgment model for a predefined alarm class has already been set. You can only set the acknowledgment model for user-defined alarm classes. Proceed as follows:

1. In the "HMI alarms" editor, click the "Alarm class" tab and select the alarm class.
2. Select the required acknowledgment model under "Properties > Properties > Acknowledgment" in the Inspector window.

Assign alarms to an alarm class requiring acknowledgment

Proceed as follows to assign an alarm to an alarm class requiring acknowledgment.

1. In the "HMI alarms" editor, click the tab for the alarm type and select the alarm.
2. Under "Properties > Properties > General" in the Inspector window, select the alarm class of the alarm.

Result

The alarm will not disappear in Runtime until it is acknowledged by the operator.

Configuring trigger for alarm acknowledgment

Introduction

You always specify the acknowledgment requirement for an alarm using the alarm class. Then the operator acknowledges the alarm using the "ACK" function key of the HMI device or the "Acknowledgment" button of the alarm view.

The following options are also available to trigger acknowledgment:

- Configuring a button to acknowledge an alarm
- Acknowledgment of a Discrete Alarm by the PLC

Requirement

- The "HMI alarms" editor is open.
- The required alarm class has been created.
- The required alarm has been created.
- An alarm view and a button are created in the "Screens" editor.

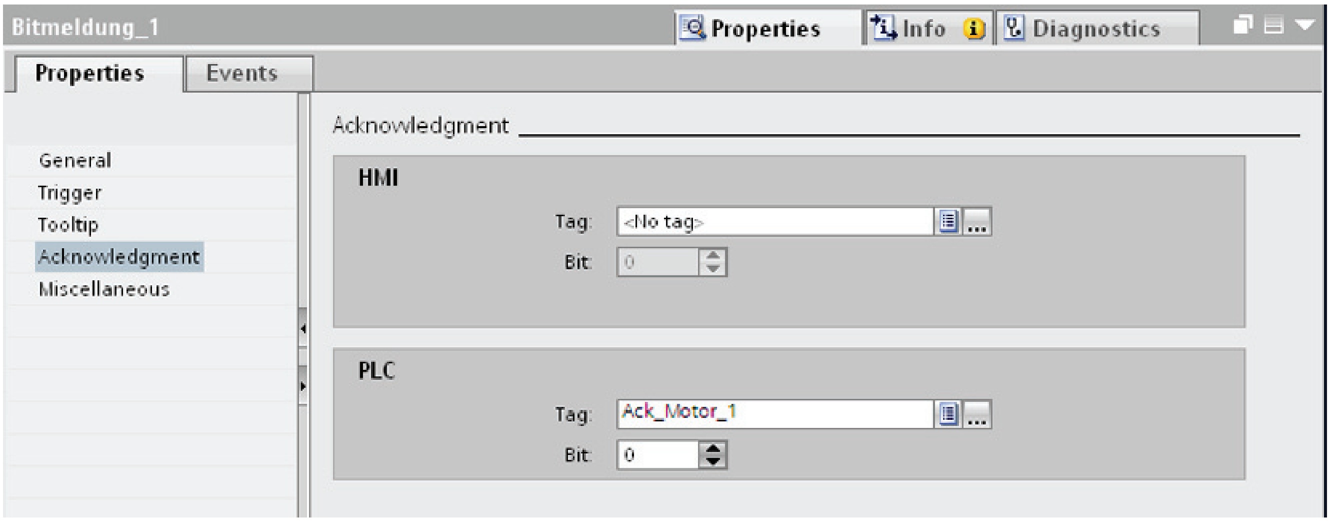
Configuring a button to acknowledge an alarm

To configure a button for acknowledging an alarm, proceed as follows:

1. Select the button in the "Screens" editor.
2. Under "Properties > Events" in the Inspector window, assign the "AlarmViewAcknowledgeAlarm" system function to the "Click" event.
3. Select the alarm view as parameter.

Acknowledgment of a Discrete Alarm by the PLC

1. In the "HMI alarms" editor, click the "Discrete alarm" tab and select the discrete alarm.
2. In the Inspector window, select the tag and the bit that acknowledges the PLC alarm under "Properties > Properties > Acknowledgment > PLC".



Sending alarm acknowledgments to the PLC

Requirement

- The "HMI alarms" editor is open.
- The required alarm has been created and assigned to an alarm class requiring acknowledgment.

Note

You cannot send the acknowledgment of analog alarms to the PLC.

Sending alarm acknowledgments to the PLC

To configure that acknowledgment of an alarm is sent to the PLC, follow these steps:

1. In the "HMI alarms" editor, click the "Discrete alarm" tab and select the discrete alarm.
2. In the Inspector window, select "Properties > Properties > Acknowledgment".
3. Under "HMI", select the tag and the bit set by the alarm acknowledgment function.

Note

Operator 舩 and PLC only have read access to the acknowledgment tag memory area.

Result

If the operator acknowledges the alarm in Runtime, the operating step is forwarded to the PLC.

10.3.3 Operating alarms in Runtime

10.3.3.1 Alarms in Runtime

Alarms

Alarms indicate events and states on the HMI device which have occurred in the system, in the process or on the HMI device itself. A status is reported when it is received.

An alarm could trigger one of the following alarm events:

- Incoming
- Outgoing
- Acknowledge
- Loop-in-alarm

The configuration engineer defines which alarms must be acknowledged by the user.

An alarm may contain the following information:

- Date
- Time
- Alarm text
- Location of fault
- Status
- Alarm class
- Alarm number
- Alarm group

Alarm classes

Alarms are assigned to various alarm classes.

- "Warnings"

  Alarms of this class usually indicate states of a plant such as "Motor switched on". Alarms in this class do not require acknowledgment.
- "Errors"

  Alarms in this class must always be acknowledged. Error alarms normally indicate critical errors within the plant such as "Motor temperature too high".
- "System"

  System alarms indicate states or events which occur on the HMI device.

  System alarms provide information on occurrences such as operator errors or communication faults.
- Custom alarm classes

  The properties of this alarm class must be defined in the configuration.

Alarm buffer

Alarm events are saved to an internal buffer. The size of this alarm buffer depends on the HMI device type.

Alarm view

The alarm view shows selected alarms or alarm events from the alarm buffer. Whether alarm events have to be acknowledged or not is specified in your configuration.

Alarm window

An alarm window shows all pending alarms or alarms awaiting acknowledgement of a particular alarm class. The alarm window is displayed as soon as a new alarm occurs.

You can configure the order in which the alarms are displayed. You can choose to display the alarms in ascending or descending order of their occurrence. The alarm window can also be set to indicate the exact location of the fault, including the date and time of the alarm event. By means of configuration, the display can be filtered in such a way that only alarms that contain a specific character string will be shown.

Alarm indicator

The alarm indicator is a graphic icon that is displayed on the screen when an alarm of the specified alarm class is activated.

The alarm indicator can have one of two states:

- Flashing: At least one unacknowledged alarm is pending.
- Static: The alarms are acknowledged but at least one of them has not gone out yet. The displayed number indicates the number of queued alarms.

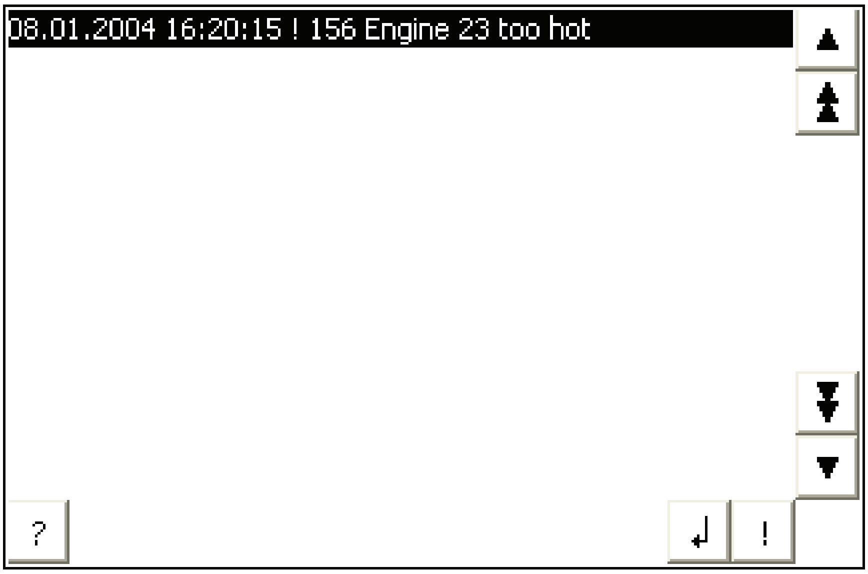
10.3.3.2 Simple alarm view, simple alarm window in runtime

Application

The simple alarm view shows selected alarms or alarm events from the alarm buffer. The layout and operation of the simple alarm window correspond to that of the simple alarm view.

Note

In the Engineering System, for example, dynamize the visibility of an object in the "Animations" tab of the Inspector window. In Runtime, the "Simple alarm view" does not support animations. If you configured an animation and, for example, run a consistency check on the project, an error alarm is displayed in the output window.



08.01.2004 16:20:15 ! 156 Engine 23 too hot

Layout

Depending on the configuration, in the alarm view different columns with information regarding an alarm or an alarm event are displayed.

To differentiate between the different alarm classes, the first column in the alarm view contains an icon:

Symbol	Alarm class
!	"Errors"
empty	"Warnings"
depends on the configuration	Custom alarm classes
$	"System"

Operation

You use the alarm view as follows, depending on how it is configured:

- Acknowledging alarms
- Editing alarms

Control elements

The buttons have the following functions:

Button	Function
!	Acknowledge alarm
↲	Loop-In-Alarm Changes to the screen that contains information about the error event
?	Displaying a tooltip for an alarm
▶	Displays the full text of the selected alarm in a separate window, namely the alarm text window In the alarm text window, you can view alarm texts that exceed the space available in the Alarm view. Close the alarm text window with the the X button.
▲	Scrolls one alarm up.
⏫	Scrolls one page up in the alarm view.
⏬	Scrolls one page down in the alarm view.
▼	Scrolls one alarm down.

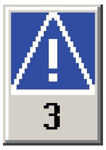
Format of the control elements

The display of the buttons for using the simple alarm view depends on the configured size. You should therefore check on the HMI device whether all the required buttons are available.

10.3.3.3 Alarm indicator in Runtime

Application

The alarm indicator is displayed if alarms of the specified alarm class are pending or require acknowledgment.



Layout

The alarm indicator can have one of two states:

- Flashing: At least one unacknowledged alarm is pending.
- Static: The alarms are acknowledged but at least one of them has not gone out yet. The displayed number indicates the number of queued alarms.

Operation

Depending on the configuration, when operating the alarm indicator an alarm window is opened. The alarm indicator can only be operated with the touch screen.

10.3.3.4 Acknowledging alarms

Introduction

You can acknowledge alarms in Runtime according to your project configuration settings. You can acknowledge alarms as follows:

- Using the display and control object buttons
- Using the "ACK" key on your HMI device
- Using individually-configured function keys or buttons

If an operator authorization is configured for an individual control, the alarms can only be acknowledged by authorized users.

To automatically acknowledge alarms in Runtime, use the system functions and the option "Acknowledgment by the PLC".

Acknowledgment variants

You acknowledge individual alarms or multiple alarms together in Runtime. They are distinguished as follows:

- Single acknowledgment

  Acknowledgment of an alarm using a button or a function key.
- Acknowledge alarm groups

  Acknowledgment of all the alarms of an alarm group using a button or a function key.

Requirement

- An alarm is displayed on the HMI device.

Procedure

To acknowledge an alarm, proceed as follows:

1. Select the alarm.
2. Click on the ! button.

Result

The alarm status changes to "Acknowledged". If the condition for triggering an alarm no longer applies, the alarm status also changes to "Outgoing", and it is no longer displayed on the HMI device.

10.3.4 Reference

10.3.4.1 System functions for alarms

System functions

System functions are predefined functions you can use to implement many tasks in runtime, even with no programming knowledge. You use system functions in a function list.

The table shows all the system functions available for displaying and editing alarms.

System function	Effect
EditAlarm	Triggers the Loop-In-Alarm event for all selected alarms.
ClearAlarmBuffer	Deletes alarms from the alarm buffer on the HMI device.

System function	Effect
ClearAlarmBufferProtoolLegacy	Function such as "ClearAlarmBuffer". This system function has been retained to ensure compatibility and uses the old ProTool numbering.
AlarmViewEditAlarm	Triggers the event Loop-In-Alarm for all alarms selected in the specified alarm view.
AlarmViewAcknowledgeAlarm	Acknowledges the alarms that are selected in the specified alarm view.
AlarmViewShowOperatorNotes	Displays the configured tooltip for the alarm selected in the specified alarm view.
AcknowledgeAlarm	Acknowledges all selected alarms.
ShowAlarmWindow	Hides or shows the alarm window on the HMI device.

10.3.4.2 System events

Basics on system events

System events

System events on the HMI device provide information about internal states of the HMI device and PLC.

The following overview illustrates when a system event occurs and how to eliminate the cause of error.

Note

HMI device dependency

Some of the system events described in this section apply to the individual HMI devices based on their scope of functions.

Note

System events are output in an alarm view. System events are output in the language currently set on your HMI device.

System event parameters

System events may contain encrypted parameters. The parameters are of relevance when troubleshooting because they provide a reference to the source code of the Runtime software. These parameters are output after the "Error code:" text.

30000 - Alarms errors when using system functions

Meaning of the system events

All system events that can be displayed are listed below. The system events are divided into different ranges.

Table 10- 2 30000 - Alarms errors when using system functions

Number	Effect/causes	Remedy
30010	The tag could not accept the function result, e.g. when it has exceeded the value range.	Check the tag types of the system function parameters.
30011	A system function could not be executed because the function was assigned an invalid value or type in the parameter.	Check the parameter value and tag type of the invalid parameter. If a tag is used as a parameter, check its value.
30012	A system function could not be executed because the function was assigned an invalid value or type in the parameter.	Check the parameter value and tag type of the invalid parameter. If a tag is used as a parameter, check its value.

40000 - Linear scaling alarms

Meaning of the system alarms

All system alarms that can be displayed are listed below. The system alarms are divided into different ranges:

Table 10- 3 40000 - Linear scaling alarms

Number	Effect/causes	Remedy
40010	The system function could not be executed since the parameters could not be converted to a common tag type.	Check the parameter types in the configuration.
40011	The system function could not be executed since the parameters could not be converted to a common tag type.	Check the parameter types in the configuration.

50000 - Data server alarms

Meaning of the system alarms

All system alarms that can be displayed are listed below. The system alarms are divided into different ranges:

Table 10- 4 50000 - Data server alarms

Number	Effect/causes	Remedy
50000	The HMI device is receiving data faster than it is capable of processing. Therefore, no further data is accepted until all current data have been processed. Data exchange then resumes.	--
50001	Data exchange has been resumed.	--

70000 - Win32 function alarms

Meaning of the system events

All system events that can be displayed are listed below.

Table 10- 5 70000 - Win32 function alarms

Number	Effect/causes	Remedy
70010	The application could not be started because it could not be found in the path specified or there is insufficient memory space.	Check whether the application exists in the specified path or close other applications.
70011	The system time could not be modified. The error alarm only appears in connection with area pointer "Date/time PLC". Possible causes: • An invalid time was transferred in the job mailbox. • The Windows user has no right to modify the system time. If the first parameter in the system event is displayed with the value 13, the second parameter indicates the byte containing the incorrect value.	Check the time which is to be set. Using Windows NT/XP: Users running WinCC Runtime must be granted the right to set the system time of the operating system.
70012	Error when executing the function "StopRuntime" with the "Runtime and operating system" option. Windows and WinCC Runtime are not closed. The error was possibly generated because other programs cannot be closed.	Close all programs currently running. Then close Windows.
70013	The system time could not be modified because an invalid value was entered. Incorrect separators may have been used.	Check the time which is to be set.
70014	The system time could not be modified. Possible causes: • An invalid time was transferred. • The Windows user has no right to modify the system time. Windows rejects the setting request.	Check the time which is to be set. Using Windows NT/XP: Users running WinCC Runtime must be granted the right to set the system time of the operating system.
70015	The system time could not be read because Windows rejects the reading function.	--

Number	Effect/causes	Remedy
70016	An attempt was made to select a screen by means of a system function or job. This is not possible because the screen number specified does not exist. Or: A screen could not be generated due to insufficient system memory. Or: The screen is blocked. Or: Screen call has not been executed correctly.	Check the screen number in the function or job with the screen numbers configured. Assign the number to a screen if necessary. Check the details for the screen call and whether the screen is blocked for specific users.
70017	Date/time is not read from the area pointer because the address set in the PLC is either not available or has not been set up.	Change the address or set up the address in the PLC.
70018	Acknowledgment that the password list has been successfully imported.	--
70019	Acknowledgment that the password list has been successfully exported.	--
70020	Acknowledgment for activation of alarm reporting.	--
70021	Acknowledgment for deactivation of alarm reporting.	--
70022	Acknowledgment to starting the Import Password List action.	--
70023	Acknowledgment to starting the Export Password List action.	--
70024	The range of values of the tag was exceeded in the system function. No calculation of the system function.	Check and correct the calculation.
70025	The range of values of the tag was exceeded in the system function. No calculation of the system function.	Check and correct the calculation.
70026	No other screens are stored in the internal screen memory. No other screens can be selected.	--
70027	The backup of the RAM file system has been started.	--
70028	The files from the RAM have been copied in the Flash memory. The files from the RAM have been copied in the Flash memory. Following a restart, these saved files are copied back to the RAM file system.	--
70029	Backup of the RAM file system has failed. No backup copy of the RAM file system has been made.	Check the settings in the "Control Panel > OP" dialog and save the RAM file system using the "Save Files" button in the "Persistent Storage" tab.
70030	The parameters configured for the system function are faulty. The connection to the new PLC was not established.	Compare the parameters configured for the system function with the parameters configured for the PLCs and correct them as necessary.
70031	The PLC configured in the system function is not an S7 PLC. The connection to the new PLC was not established.	Compare the S7 PLC name parameter configured for the system function with the parameters configured for the PLC and correct them as necessary.

Number	Effect/causes	Remedy
70032	The object configured with this number in the tab sequence is not available in the selected screen. The screen changes but the focus is set to the first object.	Check the number of the tab sequence and correct it if necessary.
70033	An e-mail cannot be sent because a TCP/IP connection to the SMTP server no longer exists. This system event is generated only at the first attempt. All subsequent unsuccessful attempts to send an e-mail will no longer generate a system event. The event is regenerated when an e-mail has been successfully sent in the meantime. The central e-mail component in WinCC Runtime attempts to connect to the SMTP server at cyclic intervals (1 minute) in order to transmit the remaining e-mails.	Check the network connection to the SMTP server and re-establish it if necessary.
70034	Following a disruption, the TCP/IP connection to the SMTP server could be re-established. The queued e-mails are then sent.	--
70036	No SMTP server for sending e-mails is configured. An attempt to connect to an SMTP server has failed and it is not possible to send e-mails. WinCC Runtime generates the system event after the first attempt was made to send an e-mail.	Configure an SMTP server: In the WinCC Engineering System using "Device settings > Device settings" In the Windows CE operating system using "Control Panel > Internet Settings > E-mail > SMTP Server"
70037	An e-mail cannot be sent for unknown reasons. The contents of the e-mail are lost.	Check the e-mail parameters (recipient etc.).
70038	The SMTP server has rejected sending or forwarding an e-mail because the domain of the recipient is unknown to the server or because the SMTP server requires authentication. The contents of the e-mail are lost.	Check the domain of the recipient address or disable the authentication on the SMTP server if possible. SMTP authentication is currently not used in WinCC Runtime.
70039	The syntax of the e-mail address is incorrect or contains illegal characters. The contents of the e-mail are discarded.	Check the e-mail address of the recipient.
70040	The syntax of the e-mail address is incorrect or contains illegal characters.	--
70041	The import of the user management was aborted due to an error. Nothing was imported.	Check your user administration or download it again to the panel.
70042	The range of values of the tag was exceeded while executing the system function. The system function was not calculated.	Check and correct the calculation.
70043	The range of values of the tag was exceeded while executing the system function. The system function was not calculated.	Check and correct the calculation.
70044	An error occurred while sending the e-mails. The e-mails were not sent.	Check the SMTP settings and the error message in the system event.
70045	Cannot load a file required for encrypting the e-mail.	Update the operating system and Runtime.

Number	Effect/causes	Remedy
70046	The server does not support encryption.	Select an SMTP server that supports encryption.
70047	The SSL versions of the HMI device and SMTP server may not be compatible.	Contact your network administrator or the operator of the SMTP server.

110000 - Offline function alarms

Meaning of the system events

All system events that can be displayed are listed below.

Table 10- 6 110000 - Offline function alarms

Number	Effect/causes	Remedy
110000	The operating mode was changed. "Offline" mode is now set.	--
110001	The operating mode was changed. "Online" mode is now set.	--
110002	The operating mode was not changed.	Check the connection to the PLCs. Check whether the address range for the “Coordination" area pointer is present in the PLC.
110003	The operating mode of the specified PLC was changed by the "SetConnectionMode" system function. The "offline" operating mode is now set.	--
110004	The operating mode of the specified PLC was changed by the "SetConnectionMode" system function. The "online" operating mode is now set.	--
110005	An attempt was made to use the "SetConnectionMode" system function to set the specified PLC to "online" mode, although the entire system is in "offline" mode. This changeover is not allowed. The PLC remains in "offline" mode.	Switch the complete system to "online" mode and repeat execution of the system function.
110006	The content of the "project ID" area pointer does not match the project ID configured in WinCC. The WinCC Runtime is therefore terminated.	Check: • the project ID entered on the PLC. • the project ID entered in WinCC.

120000 - Trend alarms

Meaning of the system alarms

All system alarms that can be displayed are listed below.

Table 10- 7 120000 - Trend alarms

Number	Effect/causes	Remedy
120000	The trend is not displayed because you configured an incorrect axis to the trend or an incorrect trend.	Change the configuration.
120001	The trend is not displayed because you configured an incorrect axis to the trend or an incorrect trend.	Change the configuration.
120002	The trend is not displayed because the tag assigned attempts to access an invalid PLC address.	Check whether the data area for the tag exists in the PLC, the configured address is correct and the value range for the tag is correct.

140000 - Connection alarms: Connection + device

Meaning of the system events

All system events that can be displayed are listed below.

Table 10- 8 140000 - Connection alarms: Connection + device

Number	Effect/causes	Remedy
140000	An online connection to the PLC is established.	--
140001	The online connection to the PLC was shut down.	--
140003	No tag update or write operations are executed.	Check whether the connection is up and the PLC is switched on. In the Control Panel, check the set parameters using the "Set PG/PC interface" function. Restart the system.
140004	No tag update or write operations are executed due to an incorrect access point, or incorrect module configuration.	Verify the connection and check whether the PLC is switched on. Check the access point or the module configuration (MPI, PPI, PROFIBUS) in the Control Panel with "Set PG/PC interface". Restart the system.
140005	No tag update or write operations are executed due to an incorrect HMI device address (possibly too high).	Use a different HMI device address. Verify the connection and check whether the PLC is switched on. Check the parameter definitions in the Control Panel using "Set PG/PC interface". Restart the system.
140006	No tag update or write operations are executed due to an incorrect baud rate setting.	Select a different baud rate in WinCC (according to module, profile, communication peer, etc.).

Number	Effect/causes	Remedy
140007	An incorrect bus profile prevents tag updates or write operations (see %1). The following parameters could not be written to the registry: 1: Tslot 2: Tqui 3: Tset 4: MinTsdr 5: MaxTsdr 6: Trdy 7: Tid1 8: Tid2 9: Gap Factor 10: Retry Limit	Check the custom bus profile. Check the connection and whether the PLC is switched on. Check the parameter definitions in the Control Panel using "Set PG/PC interface". Restart the system.
140008	An incorrect baud rate prevents tag updates or write operations. The following parameters could not be written to the registry: 0: General error 1: Wrong version 2: Profile cannot be written to the registry. 3: The subnet type cannot be written to the registry. 4: The Target Rotation Time cannot be written to the registry. 5: Incorrect Highest Station Address (HSA).	Check whether the connection is up and the PLC is switched on. In the Control Panel, check the set parameters using the "Set PG/PC interface" function. Restart the system.
140009	No tag updates or write operations because the S7 communication module was not found.	To reinstall the module, open the Control Panel and select "Set PG/PC interface".
140010	No S7 communication partner found because the PLC is shut down. DP/T: The option “PG/PC is the only master” is not set in the Control Panel under “Set PG/PC interface”.	Switch on the PLC. DP/T: If only one master is connected to the network, disable "PG/PC is the only master" in "Set PG/PC interface". If several masters are connected to the network, enable these. Do not change any settings, for this will cause bus errors.
140011	No tag updates or write operations because communication is down.	Check the connection and whether the communication partner is switched on.
140012	There is an initialization problem (e.g. if WinCC Runtime was closed in Task Manager). Or: Another application (e.g.STEP7) with different bus parameters is active and the driver cannot be started with the new bus parameters (baud rate, for example).	Restart the HMI device. Or: Start WinCC Runtime and then start your other applications.
140013	The MPI cable is disconnected and, therefore, there is no power supply.	Check the connections.
140014	The configured bus address is in use by another application.	Change the HMI device address in the PLC configuration.
140015	Incorrect baud rate Or: Incorrect bus parameters (e.g. HSA) Or: OP address > HSA or: Incorrect interrupt vector (interrupt not registered by the driver)	Correct the parameters.

Number	Effect/causes	Remedy
140016	The hardware does not support the configured interrupt.	Change the interrupt number.
140017	The set interrupt is in use by another driver.	Change the interrupt number.
140018	SIMOTION Scout disabled the consistency check. Only a corresponding note appears.	In SIMOTION Scout, reactivate the consistency check and once again download the project to the PLC.
140019	SIMOTION Scout is downloading a new project to the PLC. Connection to the PLC is canceled.	Wait until the end of the reconfiguration.
140020	The version in the PLC and in the project (FWX file) do not match. Connection to the PLC is canceled.	The following remedies are available: Download the current version to the PLC using SIMOTION Scout. Recompile the project using WinCC ES, close WinCC Runtime, and restart with the new configuration.

180000 - General alarms

Meaning of the system events

All system events that can be displayed are listed below.

Table 10- 9 180000 - General alarms

Number	Effect/causes	Remedy
180000	A component/OCX received configuration data with a version ID which is not supported.	Install a newer component.
180001	System overload because too many actions are running in parallel. Certain actions can be executed, while others are discarded.	Several remedies are available: • Generate alarms at a slower rate (polling). • Initiate scripts and functions at greater intervals. If the alarm appears more frequently: Restart the HMI device.
180002	The screen keyboard could not be activated. Possible causes: "TouchInputPC.exe" was not registered due to faulty Setup.	Reinstall WinCC Runtime.

190000 - Tag alarms

Meaning of the system events

All system events that can be displayed are listed below.

Table 10- 10 190000 - Tag alarms

Number	Effect/causes	Remedy
190000	It is possible that the tag is not updated.	--
190001	The tag is updated after the cause of the last error state has been eliminated (recovery of normal operation).	--
190002	The tag is not updated because communication with the PLC is down.	Select system function "SetOnline" to enable communication.
190004	The tag is not updated because the address configured for this tag does not exist.	Check the configuration.
190005	The tag is not updated because the configured PLC type does not exist for this tag.	Check the configuration.
190006	The tag is not updated because it is not possible to map the PLC type in the data type of the tag.	Check the configuration.
190007	The tag value is not modified because the connection to the PLC is interrupted or the tag is offline.	Set online mode or reconnect to the PLC.
190008	The configured tag limits were violated due to one of the following events: • Value input • System function • Script	Observe the configured or current tag limits.
190009	An attempt was made to assign this tag a value that is outside the valid range of values for this data type. For example, input of the value 260 for a byte tag, or input of the value -3 for an unsigned word tag.	Observe the range of values for the data type of the tags.
190010	The rate at which values are written to the tag is too high (for example, in a loop triggered by a script). Values are lost because only up to 100 operations are saved to the buffer.	The following remedies are available: • Extend the interval between multiple write actions. • Do not use an array tag longer than 6 words when configuring an acknowledgment on the HMI device using "HMI acknowledgment tag".
190011	Possible cause 1: The value entered could not be written to the configured PLC tag because the high or low limit was exceeded. The system discarded the entry and restored the original value. Possible cause 2: The connection to the PLC was interrupted.	Note that the value entered must be within the range of values of the control tag. Check the connection to the PLC.

Number	Effect/causes	Remedy
190012	It is not possible to convert a value from a source format to a target format, for example: An attempt is being made to write a counter value that is outside the valid, PLC-specific range of values. A tag of the type Integer should be assigned a value of the type string.	Check the range of values, or the data type of the tag.
190013	You entered a string that exceeds the tag length. The string is truncated automatically to a valid length.	Always enter strings that do not exceed the valid tag length.

190100 - Area pointer alarms

190100 - Area pointer alarms

Number	Effect/causes	Remedy
190100	The area pointer is not updated because the address configured for this pointer does not exist. Type 1 Warnings 2 Errors 3 PLC acknowledgment 4 HMI device acknowledgment 5 LED image 6 Trend request 7 Trend transfer 1 8 Trend transfer 2 No.: Consecutive number displayed in WinCC ES.	Check the configuration.
190101	The area pointer is not updated because it is not possible to map the PLC type to the area pointer type. Parameter type and no.: see alarm 190100	--
190102	The area pointer is updated after the cause of the last error state has been eliminated (recovery of normal operation). Parameter type and no.: See alarm 190100.	--

200000 - PLC coordination alarms

200000 - PLC coordination alarms

Number	Effect/causes	Remedy
200000	Coordination is not executed because the address configured in the PLC does not exist/is not set.	Change the address or set up the address in the PLC.
200001	Coordination is canceled because the write access to the address configured in the PLC is not possible.	Change the address or set the address in the PLC at an area which allows write access.
200002	Coordination is not carried out at the moment because the address format of the area pointer does not match the internal storage format.	Internal error
200003	Coordination can be executed again because the last error state is eliminated (return to normal operation).	--
200004	The coordination may not be executed.	--
200005	No more data is read or written. Possible causes: • The cable is defective. • The PLC does not respond, is defective, etc. • System overload	Ensure that the cable is plugged in and the PLC is operational. Restart the system if the system alarm persists.

210000 - PLC job alarms

210000 - PLC job alarms

Number	Effect/causes	Remedy
210000	Jobs are not processed because the configured address does not exist/has not been set up in the PLC.	Change the address or set up the address in the PLC.
210001	Jobs are not processed because read/write access to the configured address is not possible in the PLC.	Change the address, or set up the address in a PLC area at which read/write access is possible.
210002	Jobs are not executed because the address format of the area pointer does not match the internal storage format.	Internal error
210003	The job buffer is processed again because the last error status has been eliminated (recovery of normal operation).	--
210004	The job buffer is possibly not going to be processed.	--

Number	Effect/causes	Remedy
210005	A job mailbox with invalid number was initiated.	Check the PLC program.
210006	An error occurred while executing the job mailbox. As a result, the control job is not executed. Observe the next/previous system event.	Check the parameters of the control job. Recompile the configuration data.

220000 - WinCC communication driver alarms

220000 - WinCC communication driver alarms

Number	Effect/causes	Remedy
220001	The tag is not downloaded because write access to data type Bool/Bit is not supported by the sublevel communication driver/HMI device.	Change the configuration.
220002	The tag is not downloaded because write access to data type Byte is not supported by the sublevel communication driver/HMI device.	Change the configuration.
220003	The communication driver cannot be loaded as it is possibly not installed.	Install the driver by reinstalling WinCC Runtime.
220004	Communication is down and no update data is transferred because the cable is not connected or defective etc.	Check the connection.
220005	Communication is up.	--
220006	The connection between the specified PLC and the specified port is active.	--
220007	The connection to the specified PLC is interrupted at the specified port.	Check the following: • Is the cable plugged in? • Is the PLC OK? • Is the right port being used? • Is your configuration OK (port parameters, protocol settings, PLC address)? Restart the system if the system event persists.
220008	The communication driver cannot access or open the specified port. This port might be in use by another application, or a port that does not exist on the target device is being used. No communication with the PLC.	Close all applications that access the port and restart the computer. Use another port that exists in the system.

230000 - Screen object alarms

230000 - Screen object alarms

Number	Effect/causes	Remedy
230000	The value entered could not be used. The system discards this entry and restores the previous value. Possible causes: • The range of values is exceeded. • You entered invalid characters • The valid maximum number of users has been exceeded.	Enter a practical value, or delete a user that is no longer required.
230002	The user currently logged on does not have the necessary authorization, so the system discards the entry and restores the previous value.	Log on as user with appropriate authorization.
230003	Change to the specified screen failed because this screen is not available/configured. The current screen remains selected.	Configure the screen and check the selection function.
230005	The range of values of the tag has been exceeded in the I/O field. The original tag value is retained.	Observe the range of values for the tag when entering a value.
230100	During navigation in the Web browser, the system returned a message which may be of interest to the user. The Web browser continues to run but may not (fully) show the new page.	Navigate to another page.
230200	The HTTP channel connection was interrupted due to an error. This error is explained in detail by another system event. Data is no longer exchanged.	Check the network connection. Check the server configuration.
230201	The HTTP channel connection is set up. Data is exchanged.	--

Number	Effect/causes	Remedy
230202	WININET.DLL has detected an error. Usually, this error occurs if it is not possible to connect to the server, or the server denies a connection because the client lacks proper authorization. An unknown server certificate may also be the cause if the connection is encrypted by means of SSL. The alarm text provides details. This text is always in the language of the Windows installation because it is returned by the Windows OS. Process values are no longer exchanged. The part of the alarm returned by the Windows OS might not be displayed, e.g. "An error has occurred". WININET.DLL returns the following error: Number: 12055 Text:HTTP: <no error text available>."	Depending on the cause: When an attempt to connect fails, or a timeout occurs: • Check the network connection and the network. • Check the server address. • Check whether the WebServer is actually running on the target station. Incorrect authorization: • The configured user name and/or password do not match the entries on the server. Set consistent data. If the server certificate is rejected: Certificate signed by an unknown CA ( ): • Ignore this point, or install a certificate that has been signed with one of the root certificates known to the client station. The date of the certificate is invalid: • Ignore this point, or install a certificate with valid date on the server. Invalid CN (Common Name or Computer Name): • Ignore this point, or install a certificate with a name that corresponds to the server address.
230203	Although a connection can be made to the server, the HTTP server refused to connect. Possible causes: • WinCC Runtime does not run on the server • The HTTP channel is not supported (503 Service unavailable). Other errors can only occur if the Webserver does not support the HTTP channel. The language of the alarm text depends on the Webserver. Data is not exchanged.	On 503 Service unavailable error: Check whether WinCC Runtime is running on the server and whether the HTTP channel is supported.
230301	Internal error. An English text explains the error in more detail. The error may be caused by insufficient memory. OCX does not work.	--
230302	The name of the remote server cannot be resolved. An attempt to connect has failed.	Check the configured server address. Check whether the DNS service is available on the network.
230303	The remote server is not running on the addressed computer. incorrect server address. An attempt to connect has failed.	Check the configured server address. Check whether the remote server is running on the target computer.
230304	The remote server on the addressed computer is incompatible with VNCOCX. An attempt to connect failed.	Use a compatible remote server.

Number	Effect/causes	Remedy
230305	Authentication has failed due to incorrect password. An attempt to connect failed.	Configure the correct password.
230306	Error in the connection to the remote server. This may occur as a result of network problems. An attempt to connect failed.	Check whether the network cable is plugged in, or whether there are network problems.
230307	The connection to the remote server was shut down. Possible causes: • The remote server was shut down • The user instructed the server to close all connections. The connection is cancelled.	--
230308	This alarm provides information on the connection status. An attempt is made to connect.	--

260000 - Password system alarms

260000 - Password system alarms

Number	Effect/causes	Remedy
260000	An unknown user or an unknown password has been entered in the system. The current user is logged off from the system.	Log on to the system as a user with a valid password.
260001	The logged in user does not have sufficient authorization to execute the protected functions on the system.	Log on to the system as a user with sufficient authorization.
260002	This alarm output when the "TrackUserChange" system function is triggered.	--
260003	The user has logged off from the system.	--
260004	The user name entered into the user view already exists in the user management.	Select another user name because user names have to be unique in the user management.
260005	The entry is discarded.	Enter a shorter user name.
260006	The entry is discarded.	Use a shorter or longer password.
260007	The logoff time entered is outside the valid range from 0 to 60 minutes. The new value entered is discarded and the original value is retained.	Enter a logon timeout value between 0 and 60 minutes.
260008	An attempt was made in WinCC to read a PTProRun.pwl file created with ProTool V 6.0. Reading of the file was canceled due to incompatibility of the format.	--
260009	You have attempted to delete the user "Admin" or "PLC User". These users are fixed components of the user management and cannot be deleted.	If you need to delete a user, because perhaps you have exceeded the maximum number permitted, delete another user.

Number	Effect/causes	Remedy
260012	The password entries in the "Change Password" dialog and in the confirmation field do not match. The password is not changed. User will be logged off.	You have to log on to the system again. Then enter the identical password twice to be able to change the password.
260013	The password entered in the "Change Password" dialog is invalid because it is already in use. The password is not changed. User will be logged off.	You have to log on to the system again. Then enter a new password that has not been used before.
260014	You have tried to log on with an incorrect password three times in a row. You will be locked out and assigned to group no. 0.	You can log on to the system with your correct password. Only an administrator can change the assignment to a group.
260024	The password you entered does not meet the necessary security guidelines.	Enter a password that contains at least one number.
260025	The password you entered does not meet the necessary security guidelines.	Enter a password that contains at least one special character.
260028	Upon system start-up, an attempt to log on, or when trying to change the password of a SIMATIC log-on user, the system attempts to access the SIMATIC Logon Server. If attempting to log on, the new user is not logged in. If a different user was logged on before, then this user is logged off.	Check the connection to the SIMATIC Logon Server and its configuration; for example: 1. Port number 2. IP address 3. Server name 4. Functional transfer cable Or use a local user.
260030	The SIMATIC Logon user could not change his password on the SIMATIC Logon Server. The new password is possibly noncompliant with password rules set on the server, or the user is not authorized to change his password. The old password remains and the user is logged off.	Log in again and choose a different password. Check the password rules on the SIMATIC Logon Server.
260033	The action change password or log on user could not be carried out.	Check the connection to the SIMATIC Logon Server and its configuration; for example: 1. Port number 2. IP address 3. Server name 4. Functional transfer cable Or use a local user.
260034	The last logon operation has not yet ended. A user action or a logon dialog can therefore not be called. The logon dialog is not opened. The user action is not executed.	Wait until the logon operation is complete.
260035	The last attempt to change the password was not completed. A user action or a logon dialog can therefore not be called. The logon dialog is not opened. The user action is not executed.	Wait until the procedure is complete.

Number	Effect/causes	Remedy
260036	There are insufficient licenses on the SIMATIC Logon Sever. The logon is not authorized.	Check the licensing on the SIMATIC Logon Server.
260037	There is no license on the SIMATIC Logon Sever. A logon is not possible. It is not possible to log on via the SIMATIC Logon Server, only via a local user.	Check the licensing on the SIMATIC Logon Server.
260040	The system attempts to access the SIMATIC Logon Server upon system start-up or when trying to change the password. If attempting to log on, the new user is not logged in. If a different user was logged on before, then this user is logged off.	Check connection to the domain and its configuration in the Runtime security settings editor. Or use a local user.
260043	It was not possible to log the user on to the SIMATIC Logon Server. The user name or the password could be incorrect or the user does not have sufficient rights to log on. The new user is not logged in. If a different user was logged on before, then this user is logged off.	Try again. If necessary, check the password data on the SIMATIC Logon Server.
260044	It was not possible to log the user on to the SIMATIC Logon Server as his account is blocked. The new user is not logged in. If a different user was logged on before, then this user is logged off.	Check the user data on the SIMATIC Logon Server.
260045	The SIMATIC Logon user is not associated to any or several groups. The new user is not logged in. If a different user was logged on before, then this user is logged off.	Check user data on the SIMATIC Logon Server and the configuration in your WinCC project. A user may only be assigned to one group.

270000 - System alarms

270000 - System Alarms

Number	Effect/causes	Remedy
270000	The alarm does not indicate the tag because it is accessing an invalid address in the PLC.	Check whether the data area for the tag exists on the PLC, whether the configured address is correct, and whether the value range for the tag is correct.
270001	There is a device-specific limit as to how many alarms may be queued for viewing (see the operating instructions). This limit has been exceeded. The view no longer contains all the alarms. However, all alarms are written to the alarm buffer.	--

Number	Effect/causes	Remedy
270002	The view shows alarms of a log for which there is no data in the current project. Placeholders are output for the alarms.	Delete old log data, if necessary.
270003	The service cannot be set up because too many devices want to access this service. A maximum of four devices can execute this action.	Reduce the number of HMI devices which want to use the service.
270004	Access to the persistent alarm buffer is not possible. Alarms cannot be restored or backed up.	If the problems persist at the next restart, contact Customer Support (delete Flash).
270005	Persistent alarm buffer corrupted: Alarms cannot be restored.	If the problems persist at the next restart, contact Customer Support (delete Flash).
270006	Project modified: Alarms cannot be restored from the persistent alarm buffer.	The project was compiled and downloaded again to the HMI device. The error should no longer occur at the next restart of the HMI device.
270007	A configuration problem is preventing you from the restoring the data (e.g. a DLL was deleted, or unknown directory).	Update the operating system and download your project again to the HMI device.

290000 - Recipe system alarms

290000 - Recipe system alarms

Number	Effect/causes	Remedy
290000	The recipe tag could not be read or written. It is assigned the start value. The alarm can be entered in the alarm buffer for up to four more faulty tags. After that, alarm 290003 is output.	Check the configuration to see whether the address has been set up in the PLC.
290001	An attempt was made to assign the recipe tag a value that is outside the valid range of values for this type. The alarm can be entered in the alarm buffer for up to four more faulty tags if necessary. After that, alarm 290004 is output.	Observe the range of values for the tag type.
290002	It is not possible to convert a value from a source format to a target format. The alarm can be entered in the alarm buffer for up to four more faulty recipe tags if necessary. After that, alarm 290005 is output.	Check the range of values or type of the tag.
290003	This alarm is output after alarm 290000 was triggered more than five times. In this case, no further separate alarms are generated.	Check the configuration to see whether the tag addresses have been set up in the PLC.
290004	This alarm is output after alarm 290001 was triggered more than five times. In this case, no further separate alarms are generated.	Observe the range of values for the tag type.

Number	Effect/causes	Remedy
290005	This alarm is output after alarm 290002 was triggered more than five times. In this case, no further separate alarms are generated.	Check the range of values or type of the tag.
290006	The limits configured for the tag have been exceeded by the values entered.	Observe the configured or current tag limits.
290007	There is a difference between the source and target structure in the recipe currently being processed. The target structure contains an additional recipe tag that is not available in the source structure. The recipe tag specified is assigned its start value.	Insert the specified recipe tag into the source structure.
290008	There is a difference between the source and target structure in the recipe currently being processed.The source structure contains an additional recipe tag that is not available in the target structure and cannot be assigned. The value is discarded.	Remove the specified recipe tag in the specified recipe from the project.
290010	The storage location configured for the recipe is invalid. Possible causes: Invalid characters, write protection, data carrier out of space or not available.	Check the configured storage location.
290011	A data record of the specified number does not exist.	Check the source for the number (constant or tag value).
290012	A recipe of the specified number does not exist.	Check the source for the number (constant or tag value).
290013	An attempt was made to save a data record under a data record number that already exists. The operation is not executed.	The following remedies are available: • Check the source for the number (constant or tag value). • First, delete the data record. • Modify the "Overwrite" function parameter.
290014	The specified import file was not found.	Check the following: • The file name • Ensure that the file is in the specified directory.
290020	Check back to verify that the download of data records from the HMI device to the PLC has started.	--
290021	Check back to verify that the download of records from the HMI device to the PLC was completed.	--
290022	Check back to indicate that the download of data records from the HMI device to the PLC was canceled due to an error.	Check the following conditions in the configuration: • Are the tag addresses configured in the PLC? • Does the recipe number exist? • Does the data record number exist? • Is the "Overwrite" function parameter set?

Number	Effect/causes	Remedy
290023	Check back to verify that the download of data records from the PLC to the HMI device has started.	--
290024	Check back to verify that the download of data records from the PLC to the HMI device was completed.	---
290025	Check back to indicate that the download of data records from the PLC to the HMI device was canceled due to an error.	Check the following conditions in the configuration: • Are the tag addresses configured in the PLC? • Does the recipe number exist? • Does the data record number exist? • Is the "Overwrite" function parameter set?
290026	An attempt was made to read/write a data record that is not free at present. This error can occur if recipes were configured for download with synchronization.	Set the mailbox status to zero.
290027	Unable to connect to the PLC at present. As a result, the data record cannot be read or written. Possible causes: No hardware connection to the PLC (no cable plugged in, cable is defect), or the PLC is switched off.	Check the connection to the PLC.
290030	This alarm is output after you selected screen which contains a recipe view in which a data record is already selected.	Reload the data record from the storage location, or retain the current values.
290031	While saving, it was detected that a data record with the specified number already exists.	Overwrite the data record, or cancel the action.
290032	During the export of data records, a file with the specified name was found.	Overwrite the file, or cancel the process.
290033	Confirmation prompt before deleting data records.	--
290040	A data record error with error code %1 that cannot be described in more detail occurred. The action is canceled. It is possible that the mailbox was not installed correctly on the PLC.	Check the storage location, the data record, the "Data record" area pointer, and the connection to the PLC. Restart the action after a short waiting time. If the error persists, contact Customer Support. Forward the relevant error code to Customer Support.
290041	A data record or file cannot be saved because the storage location is out of sufficient space.	Delete files no longer required.
290042	An attempt was made to execute several recipe actions simultaneously. The last action is not executed.	Retrigger the action after a short waiting time.
290043	Confirmation prompt before saving data records.	--
290044	The database for the recipe was corrupted and will be deleted.	--
290050	A check back indicates that the export of data records was started.	--

Number	Effect/causes	Remedy
290051	A check back indicates successful completion of the export of data records.	--
290052	A check back indicates that the export of data records was canceled due to an error.	Ensure that the structure of the data records at the storage location and the current recipe structure on the HMI device are identical.
290053	A check back indicates that the import of records was started.	--
290054	A check back indicates successful completion of the import of data records.	--
290055	A check back indicates that the import of data records was canceled due to an error.	Ensure that the structure of the data records at the storage location and the current recipe structure on the HMI device are identical.
290056	Error when reading/writing the value in the specified row/column. The action was canceled.	Check the specified row/column.
290057	The tags of the recipe specified were toggled from "offline" to "online" mode. Each change of a tag in this recipe is now immediately transferred to the PLC.	--
290058	The tags of the specified recipe were toggled from "online" to "offline" mode. Modifications to tags in this recipe are no longer immediately transferred to the PLC but must be transferred there explicitly by downloading a data record.	--
290059	A check back indicates that the specified record was successfully saved.	--
290060	A check back indicates that the specified data record memory was cleared.	--
290061	A check back indicates that deletion of the data record memory was canceled due to an error.	--
290062	The data record number exceeds the maximum of 65536. This data record cannot be created.	Select another number.
290063	This occurs when you execute system function "ExportDataRecords" while the "Overwrite" parameter is set to "No". An attempt was made to save a recipe under a file name that already exists. The export is canceled.	Check the parameters of the "ExportDataRecords" system function.
290064	A check back indicates that the deletion of data records was started.	--
290065	A check back indicates successful completion of the deletion of data records.	--
290066	Confirmation prompt before deleting data records.	--
290068	Confirmation prompt for deletion of all recipe data records.	--
290069	Confirmation prompt for deletion of all recipe data records.	--

Number	Effect/causes	Remedy
290070	The data record specified was not found in the import file.	Check the source of the data record number, or the data record name (constant, or tag value).
290071	When the editing data record values, you entered a value that is below the low limit of the recipe tag. The entry is discarded.	Enter a value within the recipe tag limits.
290072	When editing data record values, you entered a value that exceeds the high limit of the recipe tag. The entry is discarded.	Enter a value within the recipe tag limits.
290073	An action (e.g. saving a record) failed for an unknown reason. The error corresponds to status alarm IDS_OUT_CMD_EXE_ERR in the large recipe view.	--
290074	While saving, a data record with the specified number but with different name was found.	Overwrite the record, change the record number or cancel the action.
290075	A data record of this name already exists. The data record is not saved.	Select a different data record name.
290110	The default values could not be set due to an error.	--
290111	The recipes subsystem cannot be used. Recipe views have no content and recipe-specific functions will not be executed. Possible causes: • Error when loading the recipes. • The recipe structure was changed in the ES. The recipes were not included in the latest project download. This means that the new configuration data no longer matches the old recipes on the device.	Download the project to the device again, including the recipes (the corresponding check box in the download dialog must be check marked).

10.4 Working with recipes

10.4.1 Basics

10.4.1.1 Definition and applications

Introduction

Related data, e.g. machine parameter assignments or production data, are combined in recipes.

Examples:

- Machine parameter settings that are needed to convert production to a different product variant.
- Product components that result in different compositions for different end products.

A recipe has a fixed data structure. The structure of a recipe is defined in the configuration. A recipe contains recipe data records. These differ in terms of their values, but not their structure.

Recipes are saved on the HMI device. A recipe data record is always transferred completely and in a single pass between the HMI device and the PLC.

Note

Restrictions in the import/export

It is not possible to export or import the recipes for Basic Panels.

Complete recipe data but not individual recipe data records can be exported and imported with ProSave to the CSV format and transmitted to the HMI device. Runtime is stopped in the meantime.

Using recipes

Recipes can be used in the following situations:

- Manual production

  You select the required recipe data and display it on the HMI device. You modify the recipe data as required and save it on the HMI device. You transfer the recipe data to the PLC.

- Automatic production

  The control program starts transfer of the recipe data between the PLC and HMI device. You can also start the transfer from the HMI device. Production is then implemented automatically. It is not essential to display or modify the data.

- Teach-in mode

  You optimize production data that was optimized manually on the system, e.g. axis positions or filling volumes. The values thus determined are transferred to the HMI device and saved in a recipe data record. You can then transfer the saved recipe data back to the PLC at a later date.

Entering and modifying the recipe data

You enter the data in the individual recipe data records and modify it as required. The following options are available:

- Data entry during configuration

  If the production data exists already, you enter the data in the "Recipes" editor during recipe configuration.

- Entering the data in Runtime

  If you have to frequently modify production data, you can do this directly in Runtime as follows:

  - Enter the data directly on the HMI device.
  - Set the parameters directly on the machine. You then transfer the data from the PLC to the HMI device and save it in the recipe.

10.4.1.2 Examples for using recipes

Recipes are used in the manufacturing industry and mechanical engineering, for example. The following recipes show typical applications which you can implement with the recipe function of WinCC:

- Machine parameter assignment

  One field of application for recipes is the assignment of machine parameters in the manufacturing industry: A machine cuts wooden boards to a certain size and drills holes. The guide rails and drill have to be moved to new positions according to the board size. The required position data are stored as data records in a recipe. You reassign the machine parameters using "Teach in" mode if, for example, a new board size is to be processed. You transfer the new position data directly from the PLC to the HMI device and save it as a new data record.

- Batch production

  Batch production in the food processing industry represents another field of application for recipes: A filling station in a fruit juice plant produces juice, nectar, and fruit drinks in a variety of flavors. The ingredients are always the same, differing only in their mixing ratios. Each flavor corresponds to a recipe. Each mixing ratio corresponds to a data record. All of the required data for a mixing ratio can be transferred to the machine control at the touch of a button.

10.4.1.3 Structure of recipes

Introduction

The basic structure of a recipe is illustrated with reference to the filling station in a fruit juice plant.

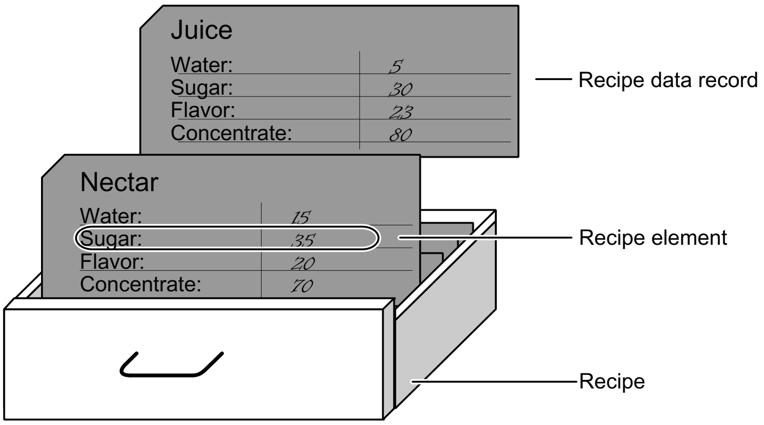
There may be several different recipes in an HMI device. A recipe can be compared to an index card box that contains several index cards. The index card box contains several variants for manufacturing a product family. All the data for each manufacturing variant is contained on a single index card.

Example:

In a soft drinks production plant, a recipe is needed for different flavors. Drink variants include fruit juice drink, juice and nectar.

Recipe

The recipe contains all the recipe data records for the different drink variants.



Recipe data records

Each index card represents a recipe data record needed to manufacture a product variant.

Recipe entries

Each index card in a drawer has the same structure. All the index cards contain fields for the different ingredients. Each field corresponds to a recipe entry. All the records of a recipe thus contain the same entries. The records differ, however, in the value of the individual entries.

Example:

All the drinks contain the following ingredients:

- Water
- Concentrate
- Sugar
- Flavoring

The records for juice drink, fruit juice or nectar differ, however, in the quantity of sugar used in production.

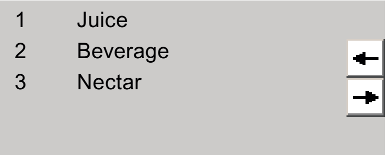
10.4.1.4 Displaying recipes

Introduction

You need to configure the recipe view to display recipes. You can change the values of a recipe in the recipe view and thereby influence the manufacturing process or a machine.

Recipe view

The recipe view is an off-the-shelf WinCC display and operator control for managing recipe data records. The recipe view is always part of a screen. The recipe view shows recipe data records in tabular form. You adapt the appearance and the possible operations to suit your specific needs.



1 Juice
2 Beverage
3 Nectar

If you are editing recipes with a recipe view in your project, the values are saved in recipe data records. The values are not transferred between the HMI device and PLC until you use the relevant operator control.

10.4.1.5 Flow of data for recipes

Interaction between the components

There is interaction between the following components at runtime:

- Recipe view

  Recipes are displayed and edited in the recipe view on the HMI device.

  The recipe data records from the internal memory of the HMI device are displayed and edited in the recipe view.

- HMI device recipe memory

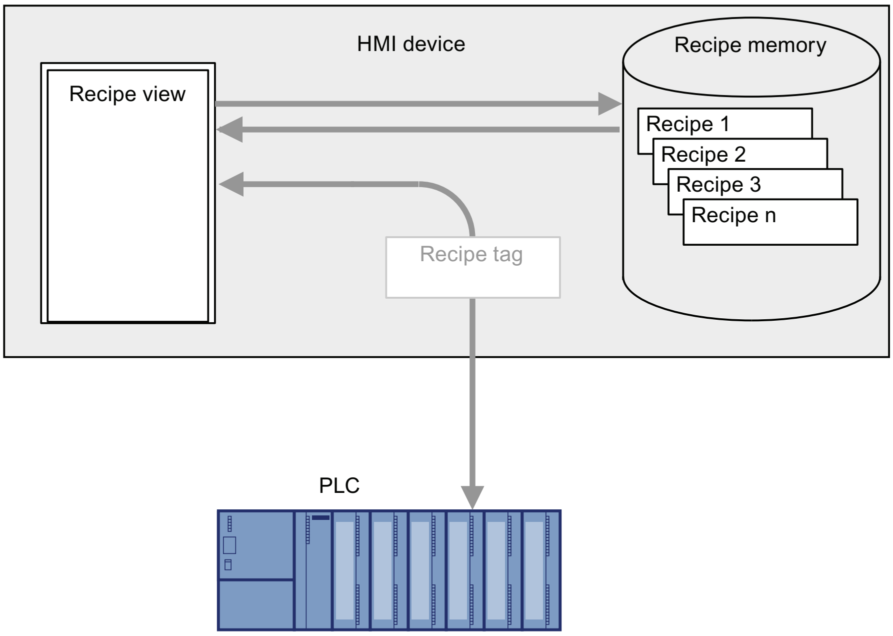
  Recipes are saved in the form of recipe data records in the HMI device's recipe memory.

- Recipe tags

  The recipe tags contain recipe data.

Overview of the flow of data

The following figure illustrates the flow of data in recipes:



To transfer recipe data records to the PLC, use the "To PLC" button in the recipe view or an operator control with the system function "RecipeViewSetDataRecordToPLC".

Data are exchanged with the PLC by recipe tags. On Basic Panels you cannot use recipe tags outside a recipe, e.g. not in I/O fields.

10.4.1.6 Synchronization of recipe data records with the PLC

Overview

When recipe data records are transferred between the HMI device and PLC, both communication peers access common communication areas on the other peer.

Recipe data records are always transferred directly. The values of the tags are written directly to or read directly from the configured addresses without being placed on the clipboard.

Data transfer types

There are two ways to transfer recipe data records between the HMI device and PLC:

- Transfer without coordination
- Coordinated transfer via the "Data record" area pointer.

> Note
>
> **Coordinated transfer**
>
> Transfer with coordinated transfer is used to prevent the uncontrolled overwriting of data in either direction in your control program.

Requirements for coordinated transfer

The following requirements apply to coordinated transfer:

- The "Data record" area pointer must be set up for the required connection in the "Communication > Connections" editor.
- In the properties of the recipe "Coordinated transfer of data records" is activated.
- The connection to the PLC is specified in the properties of the recipe with which the HMI device coordinates the transfer.

Coordinated transfer

In the case of coordinated transfer, both the PLC and the HMI device set the status bits in the shared data compartment.

Coordinated transfer of recipe data records can be a useful solution in the following cases:

- The PLC is the "active partner" for the transfer of recipe data records.
- The PLC evaluates information about the recipe number and name, as well as the recipe data record number and name.
- The transfer of recipe data records is started by the following PLC jobs:
  - "Set_data_record_in_PLC"
  - "Get_data_record_from_PLC"

10.4.2 Elements and basic settings

10.4.2.1 "Recipes" editor

Introduction

You can create, configure and edit recipes, recipe entries and recipe data records in the "Recipes" editor. The "Recipes" editor also allows you to enter values in recipe data records.

Structure of the "Recipes" editor

You create recipes in the top part of the table editor. You can also configure them there or in the Inspector window.

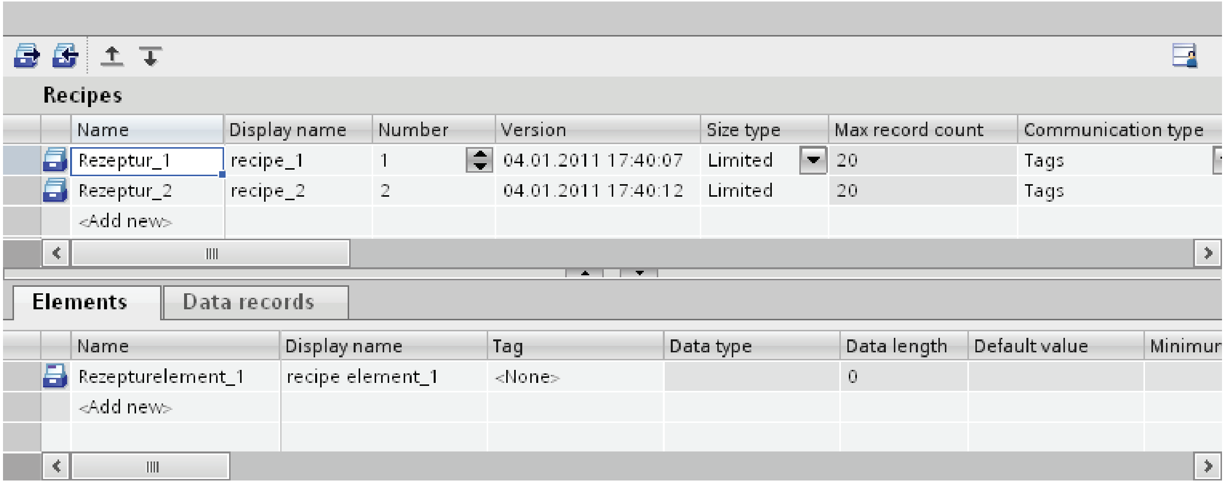
The bottom part of the table editor has the following tabs:

- Elements

  Define the recipe elements of the selected recipe using the table cells provided here. You can move recipe elements within the table with the shortcut menu commands, "Up" and "Down".

- Data records

  Define the values of the data records of the selected recipe using the table cells provided here.



Recipes

Name	Display name	Number	Version	Size type	Max record count	Communication type
Rezeptur_1	recipe_1	1	04.01.2011 17:40:07	Limited	20	Tags
Rezeptur_2	recipe_2	2	04.01.2011 17:40:12	Limited	20	Tags
<Add new>						

Elements | Data records

Name	Display name	Tag	Data type	Data length	Default value	Minimur
Rezepturelement_1	recipe element_1	<None>		0		
<Add new>						

You can then configure the selected recipe, the recipe element or the recipe data record in the Inspector window. You will find further notes on configuring the components of a recipe under "Configuring Recipes".

Recipe settings

The following settings are available for recipes:

Setting	Description
Name of the recipe	This is a unique identification for the recipe within the HMI device.
Display name	Appears in the recipe view, for example, in Runtime. You can configure display names in multiple languages. Assign descriptive names or designations which the operator can associate directly with a recipe, e.g. "fruit juice drink".
Recipe number	This is a unique identification for the recipe within the HMI device.
Version	Information about the recipe. The date and time of the last change to the recipe is set by default.
Path	Defines the storage location for recipes. The recipes are stored as a file.
Size type [fixed]	The recipe data records are limited to a predetermined number by default.

Setting	Description
Number of data records [fixed]	Maximum number of data records in a recipe in Runtime. The number is limited by the recipe memory of the HMI device.
Communication type [fixed]	The recipe data records are written directly to the addresses of the recipe tags and read from there.
Tooltip	Tooltip for the recipe which is shown to the operator in Runtime.

NOTICE

Path

The storage location depends on the storage media available on the HMI device.

Basic Panels and OP77A, TP177A (Portrait)

These HMI devices have no external memory. Recipes are always saved in the internal Flash memory. The "Path" setting is therefore not available.

Recipe element settings

You can make the following settings on the "Elements" tab:

Setting	Description
Name of the recipe element	Identifies a recipe element uniquely within the recipe. Enter meaningful names or labels that you can allocate uniquely, such as axis labels on a machine or ingredients such as "Flavoring".
Display name	Appears in the recipe view, for example, in Runtime. You can configure display names in multiple languages. Assign meaningful names or designations which the operator can associate directly, e.g. "fruit juice flavoring".
Recipe tag	An assigned tag in Runtime stores the current value of the recipe element in the recipe data record.
Data type	Data type of the recipe tag.
Data length [fixed]	Data length of the recipe tag, depending on the data type.
Text list	Text is assigned to a value or range of values in a text list. You can display this text in an output field, for example. The assigned recipe tag must have the data type of a number. The tag value must be within the range of values of the text list.
Default value	This is used as the default entry when you create a new recipe data record.
Minimum value [fixed]	The smallest representable value of a number-based recipe tag, depending on the type of data.
Maximum value [fixed]	The largest representable value of a number-based recipe tag, depending on the type of data.

Setting	Description
Decimal places	Determines how many places a decimal number is rounded to, e.g. 3 decimal places and vice versa by what power of ten an integer value is multiplied, e.g. 1,000.
Tooltip	Tooltip about the recipe element which is shown to the operator in Runtime.

Recipe data record settings

You can make the following settings on the "Data records" tab:

Setting	Description
Name of the recipe data record	Identifies a recipe data record uniquely within the recipe.
Display name	Appears in the recipe view, for example, in Runtime. You can configure display names in multiple languages. Assign meaningful names or product numbers which the operator can associate directly with a product, e.g. "yellow fruit juice E231".
Recipe data record number	Identifies a recipe data record uniquely within the recipe.
Recipe elements 1 to n	You can store various values for each recipe element even during configuration. Together with the values of the other recipe elements, a value always forms a recipe data record. You can store multiple recipe data records. If enabled in the transfer settings, the recipe data records are transferred to the HMI device when downloading the project and existing data records on the HMI device are overwritten.
Comment	Comment about the recipe data record

10.4.3 Displaying and editing recipes in Runtime

10.4.3.1 Simple recipe view

Recipe view

The simple recipe view is a ready-made display element and operator control that is used to manage recipe data records. The recipe view shows recipe data records in tabular form.

The displayed buttons and information in the columns are adjustable.

The values displayed or entered in the recipe view are saved in recipe data records. The displayed recipe data record can be written into the PLC by buttons or values can be read in from the PLC.

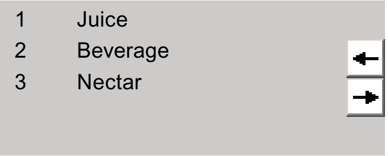
Layout of the display

The simple recipe view consists of three areas:

- Recipe list
- Data record list
- Element list

In the simple recipe view, each area is shown separately on the HMI device. Depending on the configuration, the simple recipe view starts with the recipe list.

The figure below shows an example of the data record list.



1	Juice
2	Beverage
3	Nectar

Display of values

NOTICE
Processed recipe data record is changed in the background Only applies for Basic Panels: if an operator has changed a recipe data record and a PLC job wants to read or write any recipe data record of this recipe, the PLC job is stopped and a system alarm is output. On the other hand, the changed value is displayed immediately if only the PLC job and no operator has changed recipe data. Does not apply for Basic Panels: If an operator has changed a recipe data record and a PLC job has changed the values of the recipe data record concerned, the recipe view is not updated automatically. To update the recipe view, reselect the respective recipe data record.

See also

Recipe view (Page 2105)

10.4.3.2 Behavior of the recipe view in Runtime

Screen change

If you change to another screen and have not yet saved changes to the recipe data in the recipe view, you will be prompted to save the recipe data. The recipe name and the name of the recipe data record are displayed to show which recipe data have not been saved yet.

Create, change, copy or delete recipe data records

If you attempt to create a new recipe data record and a recipe data record already exists, a system alarm will appear on screen.

Operating the recipe view with function keys

The Recipe view can be operated with function keys, e.g. if the HMI device does not have touch functionality. You can assign functions such as "SaveDataRecord" to the function keys on the HMI device.

Display after import of recipe data

NOTICE
Availability Import and export of recipe data is not available for Basic Panels and OP77A, TP177A (Portrait).

If you open the recipe view during the import of recipe data, only the recipe data that is already completely imported will be displayed. The recipe view is not automatically updated with a data import. In order to have a complete view of all the recipe data, do not open the recipe view until the system prompts you that the recipe data has been imported successfully. Alternatively, update the recipe view after successful completion of the import procedure.

Updating tag for recipes and recipe data records

NOTICE
Availability Tags for recipes and recipe data records are not available for Basic Panels and OP77A, TP177A (Portrait).

The current recipe data record or its number can be saved to a tag, depending on the configuration. The tag will be updated under the following conditions:

- The recipe data record has been loaded.
- The screen with the recipe view was not exited during loading.

This operation may take some time.

10.4.4 Configuring recipes

10.4.4.1 General configuration procedure

Carry out the following configuration steps when you create a new recipe:

Step	Description
1	Define the structure of the recipe.
2	Create tags according to the recipe structure. Assign process names to these tags.
3	Create the recipe.
4	Enter the required properties for the recipe: • Language-dependent view name of the recipe • "Coordinated transfer of data records" option Not for Basic Panels: • Recipe storage location • "Synchronize recipe view and recipe tags" option • "Manual transfer of individual modified values (teach-in mode)" option
5	Create the recipe elements and enter the required properties: • Language-dependent view name of the recipe elements • Tag binding of the recipe elements • Standard values and decimal places (power of ten) for the recipe elements
6	Create the recipe data records. Enter the language-specific display names for the recipe data records.
7	Configure a screen with recipe view or a recipe screen.

NOTICE

Basic Panels and OP77A, TP177A (Portrait)

The selection of the storage location is not available for these devices. The recipes are always saved in the internal Flash memory.

Recipe tags cannot be used outside a recipe, e.g. not in I/O fields, not in alarms as trigger tags, not in systems functions as parameters, etc.

NOTICE

Restrictions recipe view and recipe image

Only the simple recipe view is available in Basic Panels and OP77A, TP177A. Recipe images are not available in Basic Panels and OP73, OP77A, TP177A (Portrait).

10.4.4.2 Creating and Editing Recipes

Creating a new recipe

Introduction

To create a complete recipe, start by creating a new recipe, assign the corresponding recipe elements and then define the associated values in a recipe data record.

Requirement

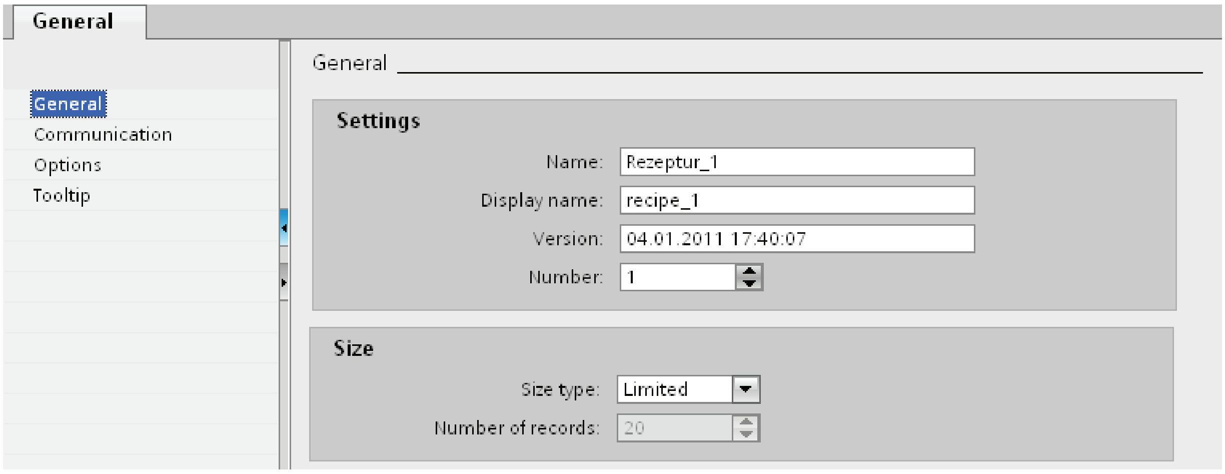
- The tags for the recipe have been created.
- The "Recipes" editor is open.

Create recipe

Create a recipe as follows:

1. Click "Add" in the first free row of the table in the "Recipes" editor.

   The new recipe is created and displayed on a line.



Settings

Name: Rezeptur_1

Display name: recipe_1

Version: 04.01.2011 17:40:07

Number: 1

Size

Size type: Limited

Number of records: 20

2. Enter a descriptive name for the recipe under "Name" in the "General" area.

   This name identifies the recipe unambiguously within the project.

3. Select "Display name" to enter the language-specific name to be displayed in runtime.
4. Select a recipe number in "Number".

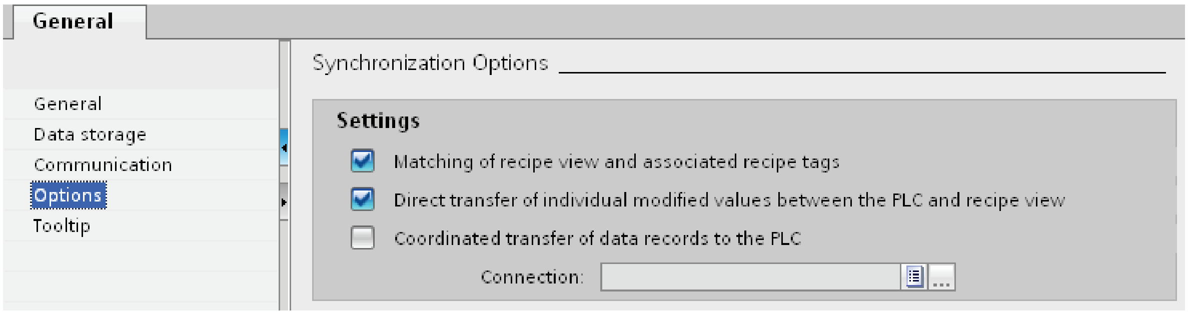
   The number identifies the recipe unambiguously within the HMI device.

   The recipe is automatically assigned a version that indicates the date and time of the last change. As an alternative, you can enter specific information relating to the recipe.

5. Specify the storage location for recipe data records in "Data medium". The options offered depend on the specific HMI device used.

> NOTICE
>
> **Basic Panels and OP77A, TP177A (Portrait)**
>
> The selection of the storage location is not available for these devices. The recipes are always saved in the internal Flash memory.
>
> Recipe tags cannot be used outside a recipe, e.g. not in I/O fields, not in alarms as trigger tags, not in systems functions as parameters, etc.

6. Enter a tooltip that is shown to the operator in runtime.
7. To compare recipe tags which are configured in I/O fields with the recipe view in Runtime, activate "Synchronize recipe view and recipe tags" in the Inspector window under "Properties > Synchronization".



Settings

☑ Matching of recipe view and associated recipe tags

☑ Direct transfer of individual modified values between the PLC and recipe view

□ Coordinated transfer of data records to the PLC

Connection:

> NOTICE
>
> **Basic Panels and OP77A, TP177A (Portrait)**
>
> Because the recipe tags cannot be additionally used in I/O fields in screens for Basic Panels, the "Synchronize recipe view and recipe tags" is not available; you will also not be able to use the "Manual transfer of individual modified values (teach-in mode)" option.

8. Deactivate "Manual transfer of individual modified values (teach-in mode)" to specify that the recipe tags are automatically transferred to the PLC when editing the I/O fields.
9. Activate "Coordinated transfer of data records" to monitor the transfer of recipe data in Runtime using area pointers.
10. Select the appropriate connection to the PLC for coordinated transfer under "Synchronize with".

Create recipe element

To create recipe elements, proceed as follows:

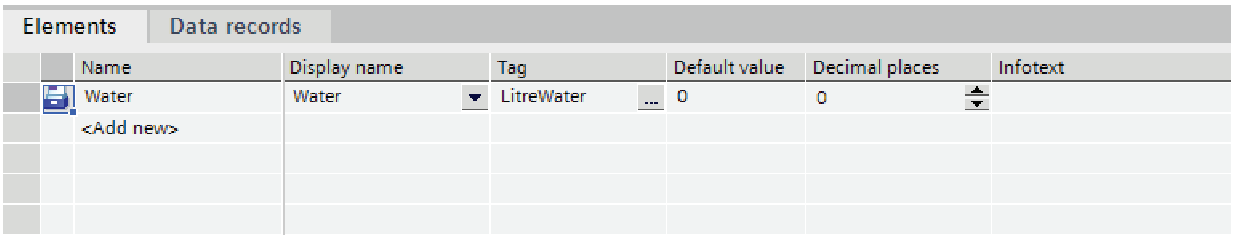
1. Click the "Elements" tab.
2. Click "Add" in the first free line of the table editor.

   A new recipe element is created.
3. Enter a descriptive name for the element under "Name".

   The name identifies the element uniquely within the recipe.
4. Enter a language-specific display name for the element under "Display name".

   The display name appears in the recipe view, for example, in runtime.
5. Select the tag you want to link to the recipe element under "Tag".

   The value of the recipe data element is saved in Runtime in this tag, which is stored in a recipe data record.

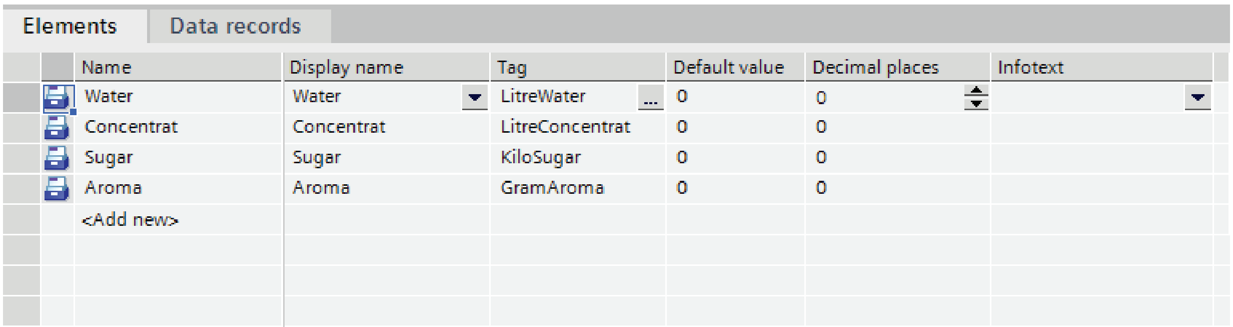


Name	Display name	Tag	Default value	Decimal places	Infotext
Water	Water	LitreWater	0	0	
<Add new>					

6. Enter a tooltip.

   The tooltip is shown to the operator in Runtime.
7. Under "Default value", enter the value that you want to use as the default entry when you create a new recipe data record.
8. To assign text to a value or range of values, select the relevant text list here. The assigned recipe tag must have the data type of a number. The tag value must be within the range of values of the text list.

   The text stored in the text list is displayed in an output field, for example, in Runtime.
9. Determine exactly how many places a decimal number is rounded to in the "Decimal places" column, e.g. 3 decimal places and vice versa by what power of ten an integer value is multiplied, e.g. 1,000.
   Examples for 3 decimal places: Entering "5" for a recipe element with the "Integer" data type gives the value "5000". Entering "5.6789" for a recipe element with the "Real" data type gives the value "5.679".
10. Create as many recipe entries as needed for the recipe. The maximum number of recipe entries possible depends on the HMI device being used.



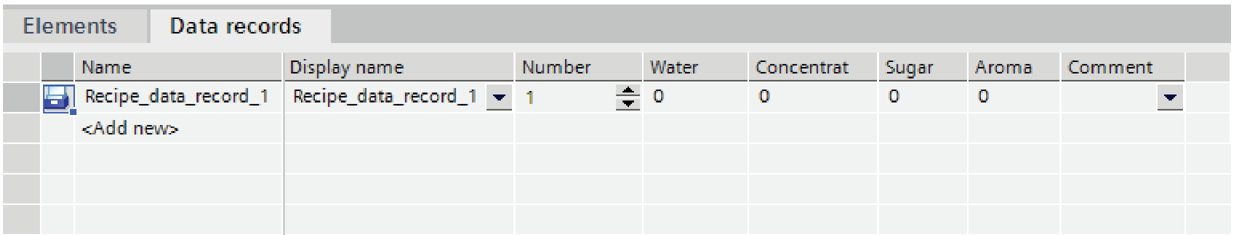
Name	Display name	Tag	Default value	Decimal places	Infotext
Water	Water	LitreWater	0	0	
Concentrat	Concentrat	LitreConcentrat	0	0	
Sugar	Sugar	KiloSugar	0	0	
Aroma	Aroma	GramAroma	0	0	
<Add new>					

Create recipe data record with known recipe values

To create recipe elements, proceed as follows:

1. Click the "Data records" tab.
2. Click "Add" in the first free line of the table editor.

   A new recipe data record is created. The recipe data record has a separate column for every recipe element created in the recipe.



Name	Display name	Number	Water	Concentrat	Sugar	Aroma	Comment
Recipe_data_record_1	Recipe_data_record_1	1	0	0	0	0	
<Add new>							

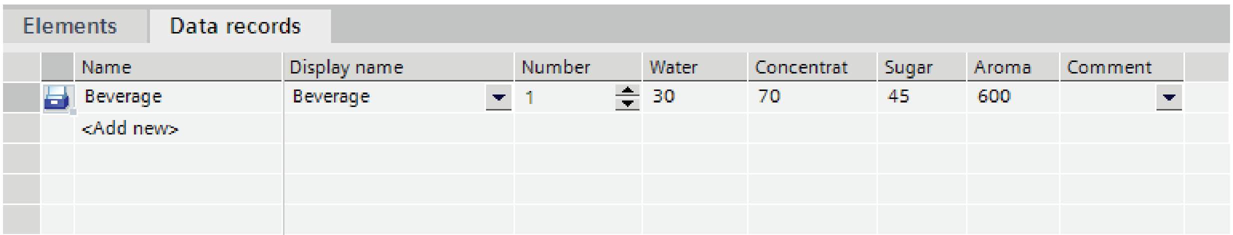
3. Enter a descriptive name under "Name".

   The name identifies the data record uniquely within the recipe.
4. Enter a language-specific name under "Display name".

   The display name appears in the recipe view, for example, in runtime.
5. Enter a recipe data record number under "Number".

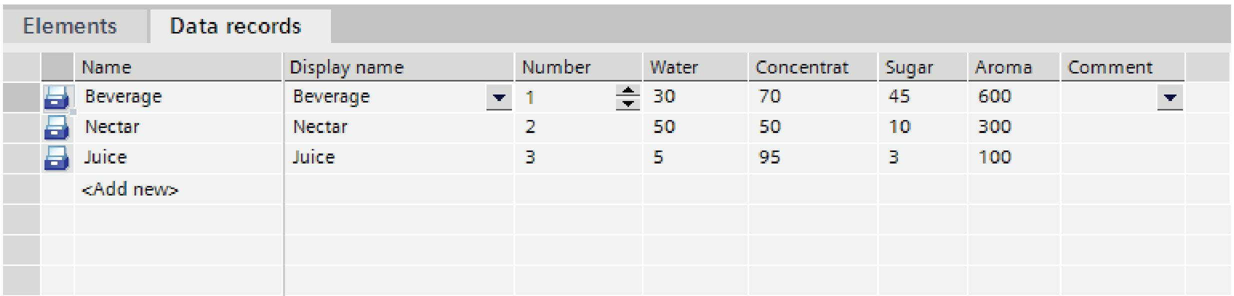
   The recipe data record number identifies the recipe data record uniquely within the recipe.

6. If you already know the recipe values at the configuration stage, you can enter the relevant value for each recipe element.



Name	Display name	Number	Water	Concentrat	Sugar	Aroma	Comment
Beverage	Beverage	1	30	70	45	600	
<Add new>							

7. Create as many data records as you need for the recipe.



Name	Display name	Number	Water	Concentrat	Sugar	Aroma	Comment
Beverage	Beverage	1	30	70	45	600	
Nectar	Nectar	2	50	50	10	300	
Juice	Juice	3	5	95	3	100	
<Add new>							

Enter the values in runtime

The following options are available for entering values in the recipe data records at runtime:

- Transfer data directly from the PLC (Teach-in mode)
- Import of values from a CSV file
- Input values on the HMI device

NOTICE
Basic Panels and OP77A, TP177A (Portrait) The import of values is not available for these devices.

Result

The complete recipe is configured.

Recipe data records with date or time stamp

If you use date or time data, make sure that the system setting for time and date on the configuring computer match those on the target system. Example: You load a recipe data record on the target system at 13:55 in which 14 h is stored as the processing time. If it is already 14:05 on the target computer, the recipe will not be processed. If an operator processes the recipe, change information will not written back correctly into the database.

After loading to the target system, check the recipes with date or time stamps on the target system.

Editing a recipe

Purpose

You want to modify, extend or delete parts of a recipe.

Requirement

- You have created at least one recipe.
- The "Recipes" editor is open.

Changing recipe settings

To change the recipe settings, proceed as follows:

1. Select the recipe that you want to change in the "Recipes" editor.

   The Inspector window opens.
2. Change the recipe configuration in the Inspector window.

You change recipe elements and recipe data records in the same way.

Change recipe values

To change recipe values, proceed as follows:

1. Select the recipe whose values you want to change.
2. Click the "Data records" tab.
3. Enter the new values in the value columns.

Adding a recipe element

To add more recipe elements to a recipe, proceed as follows:

1. Select the recipe to which you want to add more elements in the "Recipes" editor.
2. Click the "Elements" tab.
3. Click "Add" in the first free line.

   The recipe element is created.
4. Configure the recipe element.

You add recipe data records in the same way.

Managing recipes

Requirement

- You have created a recipe with recipe elements and recipe data record.
- The "Recipes" editor is open.

Renaming recipes

We distinguish between internal names and display names for recipes, recipe entries and recipe data records.

To rename recipe elements, proceed as follows:

1. Select the recipe that you want to rename.

   The Inspector window opens.
2. Select the "Rename" command from the shortcut menu.
3. Enter the new name.

   You rename recipe elements and recipe data records on the relevant tab in the same way.

   Note

   The view names in the "Recipes" editor can also be renamed under "Languages & Resources > Project Texts". This possibility is useful when you have already configured in several languages for example.

Copying and pasting recipes

To copy and paste recipes, proceed as follows:

1. Select the recipe that you want to copy.
2. Select the "Copy" command from the shortcut menu.
3. Select the "Paste" command from the shortcut menu in the first free table row.

The copied recipe is pasted into the table. The recipe elements and recipe data records are also copied in the appropriate tab with the recipe.

You also copy the recipe elements and recipe data records on the appropriate tab in the same way.

If a recipe data record of the same name already exists, the name of the copied recipe data record is extended by one digit. This ensures that the name is unique. Recipe data records can only be copied or pasted within the same recipe.

Deleting a recipe

To delete a recipe, proceed as follows:

1. Select the recipe that you want to delete.
2. Select the "Delete" command from the shortcut menu.

   The recipe is deleted.

You delete recipe elements and recipe data records on the relevant tab in the same way.

Note

When a recipe is deleted, the recipe data records contained in the recipe are also deleted.

Note

When you delete a recipe element, the associated values in the recipe data records are also deleted. The assigned tags are retained.

10.4.4.3 Configuring the display of recipes

Configuring the simple recipe view

Requirement

- You have created the recipe.
- The "Screens" editor is open.
- The screen has been created and opened.

CAUTION

Data loss with several recipe views in the screen

Applies only to Basic Panels, OP73, OP77A, TP177A and TP177A (Portrait): If two or more recipe views show the same recipe in a screen, you have a conflict when accessing the data.

The result is data loss and unpredictable status of recipe data.

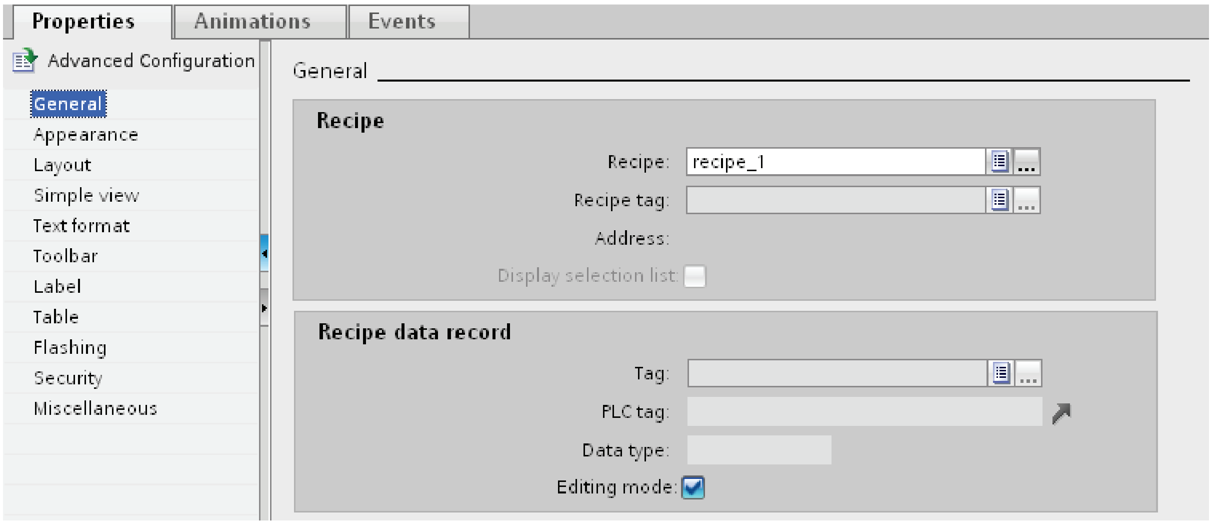
Make sure the operators do not select and edit the same recipe in different recipe views.

- Display only one recipe in a recipe view.
- Display a different recipe in each recipe view.

Procedure

To configure a simple recipe view, proceed as follows:

1. Paste the recipe view into the screen. You will find the recipe view under "Controls" in the "Tools" task card.
2. Only in devices which also support the extended recipe view: Activate "Simple view" under "Properties > Display > Mode".
3. If you want to display only the recipe data records of a specific recipe in the recipe view, select the specific recipe under "Properties > General > Recipe".



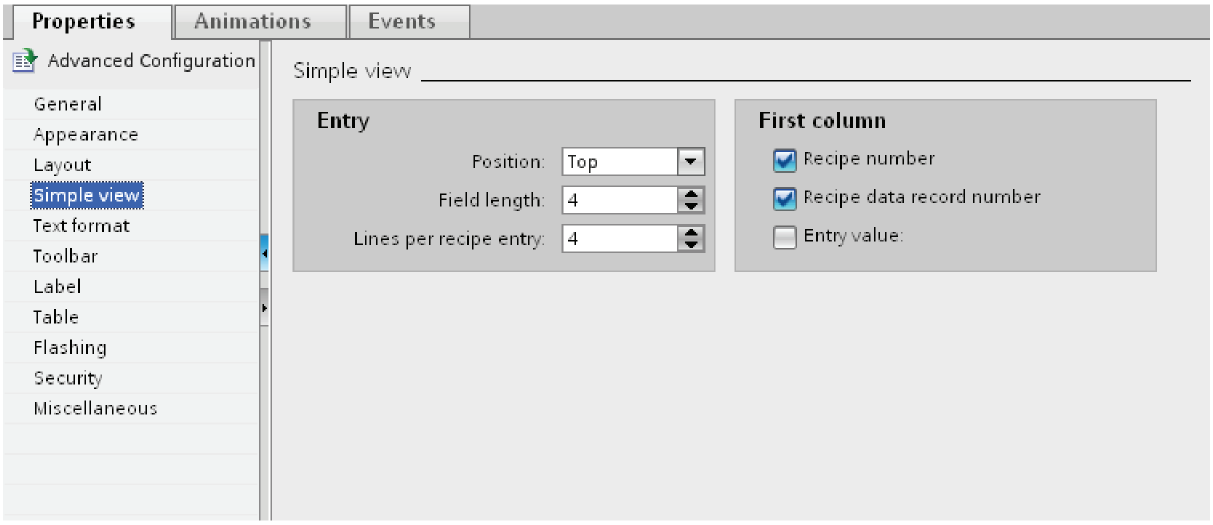
Recipe: recipe_1

Editing mode: ☑

4. If you only want to display the recipe data in the recipe view, deactivate "Processing mode" in the "Recipe data record" area.
5. You can define additional options for the recipe view under "Properties > Appearance" and "Properties > Layout".
6. Select "Properties > Simple view" to select the position, the field length, and the number of lines required.

   Select "Position > Top" to display the recipe value in the first line of the recipe entry.

   Select "Position > Bottom" to display the recipe value in the last line of the recipe entry.



Properties | Animations | Events

Advanced Configuration

- General
- Appearance
- Layout
- Simple view
- Text format
- Toolbar
- Label
- Table
- Flashing
- Security
- Miscellaneous

Simple view

Entry

Position: Top

Field length: 4

Lines per recipe entry: 4

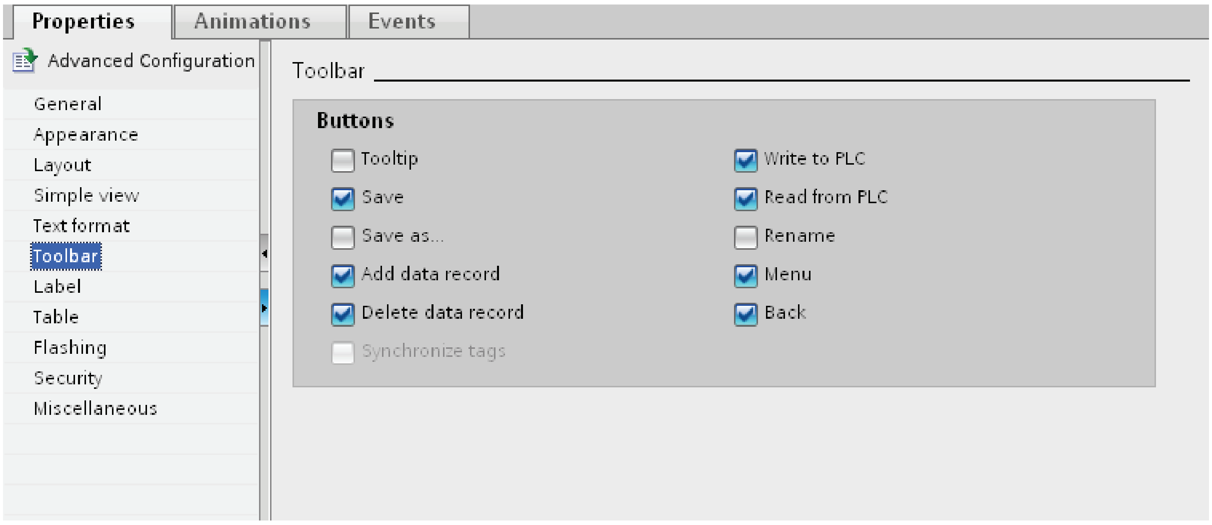
First column

☑ Recipe number

☑ Recipe data record number

□ Entry value:

7. Under "Properties" > Toolbar" specify which menu commands are available in the recipe view in Runtime.



Properties | Animations | Events

Advanced Configuration

- General
- Appearance
- Layout
- Simple view
- Text format
- Toolbar
- Label
- Table
- Flashing
- Security
- Miscellaneous

Toolbar

Buttons

□ Tooltip

☑ Save

□ Save as...

☑ Add data record

☑ Delete data record

□ Synchronize tags

☑ Write to PLC

☑ Read from PLC

□ Rename

☑ Menu

☑ Back

Result

The simple recipe view is configured. You can use the recipe view to display and edit recipe data during runtime.

10.4.5 Using recipes in Runtime

10.4.5.1 Using the simple recipe view

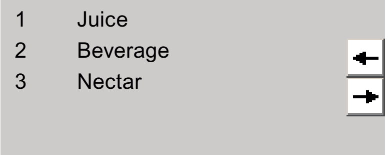
Description of the simple recipe view

Layout

The simple recipe view consists of the following display areas:

- Recipe list
- Data record list
- Element list

This application is illustrated below:



1 Juice
2 Beverage
3 Nectar

In the simple recipe view, each area is shown separately on the HMI device. You can use the shortcut menu to operate each of these display areas.

The simple recipe view always begins with the recipe list.

Operation

You have the following options for using the simple recipe view, according to the configuration:

- Create, change, copy or delete recipe data records
- Read recipe data records from the PLC or transfer to the PLC

Using the display area and shortcut menu

Toggle between the display areas and the shortcut menus to operate the simple recipe views.

The table below shows the operation of the display area.

Button	Key	Function
	<Enter>	The next lowest display area is opened, i.e. the data record list or the element list.
←	<Esc>	The previous display area opens.

Button	Key	Function
	<INS>	Creates a new data record for the selected recipe if the list of recipes or recipe data records is displayed. Then changes to the list of recipe element. Requirement: "Properties >General > Processing mode" is activated. The button can be simulated with the "Key SimulateSystemKey" function even on devices without keys.
	<DEL>	Deletes the selected recipe data record in the list of recipe data records. Requirement: "Properties >General > Processing mode" is activated.
	<Up>/<Down>	Selects the previous/next entry.
	<Pg Up>/<Pg Down>	Moves the display up or down one page.
	<Home>/<End>	Selects the first/last entry. The first/last entry is selected.

The table below shows the operation of the shortcut menu:

Button	Key	Function
→	<Right>	The shortcut menu of the display area opens.
←	<Esc>	The menu is closed. The display area opens.
	Input of the number of the menu command	The menu command is executed.

Shortcut menus of the simple recipe view

You can click the → button in each display area to call up a selection of commands. The command selection lists those commands that are available in the current display area. A number is assigned to each command. The command is executed when you enter this number. Alternatively select the command and press the <Return> key.

Shortcut menus in the recipe list

Menu command	Function
New	A new recipe data record is created for the selected recipe. If a start value is configured, it is displayed in the input field.
Display tooltip	The tooltip configured for the recipe is displayed.
Open	The record list of the selected recipe opens.

Shortcut menus of the recipe data record list

Menu command	Function
New	Creates a new recipe data record. If a start value is configured, it is displayed in the input field.
Deleting	The displayed record is deleted.
Save as	The selected data record is saved under a different name. A dialog box opens where you can enter the name.
Rename	Renames the selected data record. A dialog box opens where you can enter the name.
Open	The element list of the selected data record opens.
Previous	The recipe list opens.

Shortcut menus of the recipe element list

Menu command	Function
Save	The selected data record with the recipe element is saved.
To PLC	The displayed values of the selected data record are transferred from the HMI device to the PLC.
From PLC	The recipe values from the PLC are displayed in the recipe view of the HMI device.
Save as	The data record is saved under a new name. A dialog box opens where you can enter the name.
Display tooltip	The tooltip configured for the recipe element is displayed.
Rename	The selected recipe element is renamed. A dialog box opens where you can enter the name.
Previous	The data record list opens.

Shortcut menus in the data record list

Note

HMI device dependency

The following menu commands are configured in Basic Panels and in OP 77A, TP 177A, TP 177A (Portrait) and TB 177B.

Menu command	Function
To PLC	The displayed values of the selected data record are transferred from the HMI device to the PLC.
From PLC	The recipe values from the PLC are displayed in the recipe view of the HMI device.

Managing recipe data records

Recipe data record administration

You have the following options for managing the simple recipe view, according to the configuration:

- Creating new recipe data records
- Copy recipe data records
- Edit recipe data records
- Delete recipe data records

Creating new recipe data records

To create a new recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to create a new recipe data record.
2. Select the "New" command from the shortcut menu for the recipe list.

   A new data record with the next available number will be created.

   The element list of the new recipe data record opens.
3. Enter values for the elements of the recipe data record.

   The configuration data may already contain default values for the recipe data record.
4. Select the "Save" command from the shortcut menu for the element list.

   The dialog "Save as" opens.
5. Enter the name and number of the recipe data record.
6. Click the "OK" button.

Result

The new recipe data records will be saved to the selected recipe. If the recipe data records already exists, a system event will be output to the screen.

Copying a recipe data record

To copy a recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to copy an existing recipe data record.
2. On the HMI device, select the recipe data record of which you want to save a copy.
3. Select the "Save As" command from the shortcut menu for the data record list.

   The dialog "Save as" opens. The recipe data record is automatically given the next free recipe data record number.
4. Under name, enter the name of the record.
5. Click the "OK" button.

Result

The recipe data record is stored under the new name.

Modify recipe data record

To change a recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to edit an existing recipe data record.
2. Select the recipe data record that you want to edit on the HMI device.
3. Select the recipe data record.

   The element list of the recipe data record is displayed.
4. Replace the old values with new ones.
5. Select the "Save" command from the shortcut menu for the element list.

Result

The modified values are applied to the recipe data record.

Deleting a recipe data record

To delete a recipe data record, proceed as follows:

1. Select the recipe on the HMI device from which you want to delete an existing recipe data record.
2. Select the recipe data record that you want to delete on the HMI device.
3. Select the "Delete" command from the shortcut menu for the data record list.
4. Confirm this security prompt to delete the data record.

Result

The recipe data record is deleted.

Read recipe data record from PLC

Introduction

In Runtime, you can change values directly in the plant that are also stored in recipes in the HMI device. This applies if a valve was opened further directly in the plant than was specified in the recipe. The values of the recipe data records saved in the HMI device possibly no longer match the values in the PLC.

You can read the values of the recipe tags from the PLC and write them to a recipe data record.

The read values are written to the recipe data record that is currently displayed on the HMI device.

Procedure

To read a recipe data record from the PLC, proceed as follows:

1. Open the recipe on the HMI device.

   The data record list opens.
2. Select the element list of the recipe data record to which you want to apply the values from the PLC.
3. Select the "From PLC" command from the shortcut menu for the element list.

   The values are read from the PLC and displayed in the current recipe data record.
4. If you want to save the values, select the "Save" or "Save As" command.

Result

The values are read from the PLC, visualized on the HMI device and saved to the recipe data record.

Transferring a recipe data record to the PLC

Introduction

For the values of a data record that was changed in the recipe view to take effect, you must transfer the values to the PLC.

The values displayed in the recipe view are always transferred to the PLC.

Procedure

To transfer a recipe data record to the PLC, proceed as follows:

1. Open the recipe you want to use.

   The data record list opens.

2. Select the element list of the recipe data record whose values you want to transfer to the PLC.
3. Select the "To PLC" command from the shortcut menu for the element list.

Result

The values of the recipe data record are transferred to the PLC.

10.4.6 Example

10.4.6.1 Example of creating a recipe

Task

In this example, you create three recipes for a fruit juice mixing machine. The fruit juice mixing machine produces drinks with "orange", "apple" and "tropical" flavors. You create a recipe for each flavor.

Each recipe contains a recipe data record for the following mixing ratios:

- Beverage
- Nectar
- Juice

Settings

The settings relate to an HMI device which is connected to a SIMATIC S7-300 or SIMATIC S7-400.

In this example, you will need the following tags, recipes, recipe entries and recipe data records:

Tags:

Name	PLC connection	Address	Type
Liter water	Yes	DB 120, DBW 0	Integer
Liter concentrate	Yes	DB 120, DBW 4	Integer
Kilo sugar	Yes	DB 120, DBW 8	Integer
Gram flavoring	Yes	DB 120, DBW 12	Integer

Recipes:

- Orange
- Apple
- Tropical

Recipe entries:

Recipe element	Associated tag
Liter water	Liter water
Liter concentrate	Liter concentrate
Kilo sugar	Kilo sugar
Gram flavoring	Gram flavoring

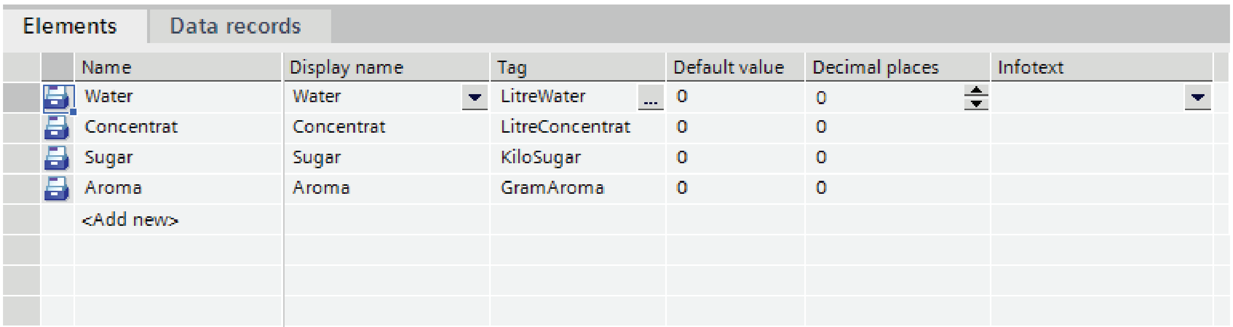
Recipe data records for drink, nectar and juice:

Data record name	Liter water	Liter concentrate	Kilo sugar	Gram flavoring
Beverage	30	70	45	600
Nectar	50	50	10	300
Juice	5	95	3	100

Procedure

To create a recipe, proceed as follows:

1. Create the following tags with the settings specified above: "LiterWater", "LiterConcentrate", "KiloSugar" and "GramFlavoring".
2. Create the "Orange", "Apple" and "Tropical" recipes with the settings indicated above. Create the recipe entries in each recipe.



Name	Display name	Tag	Default value	Decimal places	Infotext
Water	Water	LitreWater	0	0	
Concentrat	Concentrat	LitreConcentrat	0	0	
Sugar	Sugar	KiloSugar	0	0	
Aroma	Aroma	GramAroma	0	0	
<Add new>					

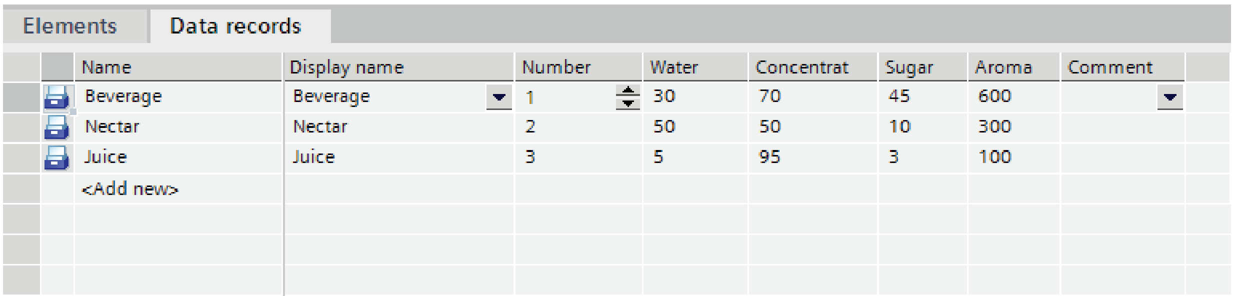
3. Not for Basic Panels: Configure each recipe so that you can synchronize the recipe data records between the recipe screen and recipe view. The values of the recipe tags should not be transferred automatically to the PLC.

You will have to make the following settings in the Properties dialog for the recipe concerned:

Under "Properties > Options":

- Activate the "Synchronize recipe view and recipe tags" option.
- Activate the "Manual transfer of individual modified values (teach-in mode)" option.

4. Create the data records indicated above in each recipe. Enter the values indicated above in each of the data records.



Name	Display name	Number	Water	Concentrat	Sugar	Aroma	Comment
Beverage	Beverage	1	30	70	45	600	
Nectar	Nectar	2	50	50	10	300	
Juice	Juice	3	5	95	3	100	
<Add new>							

Result

The "Orange", "Apple" and "Tropical" recipes have been created.

10.5 Configuring user administration

10.5.1 Field of application of the user administration

Principle

The access protection controls access to data and functions in Runtime. This feature protects your applications against unauthorized operation. Safety-related operations are already limited to specific user groups when a project is being created. To this purpose you set up users and user groups that you equip with characteristic access rights, so-called authorizations. You then configure the authorizations required for operation of safety-related objects. Operators only have access, for example, to specific operator controls. Commissioners, for example, have unlimited access in Runtime.

Definition

You administer users, user groups and authorizations centrally in the user administration of WinCC. You transfer users and user groups together with the project to the HMI device. The users and passwords are managed on the HMI device in the User view.

Application example

You configure the "Service" authorization so that only service technicians have access to the configuration parameters. You assign the authorization to the "Service technician" user group. This allows all members of this group to set the protected configuration parameters.

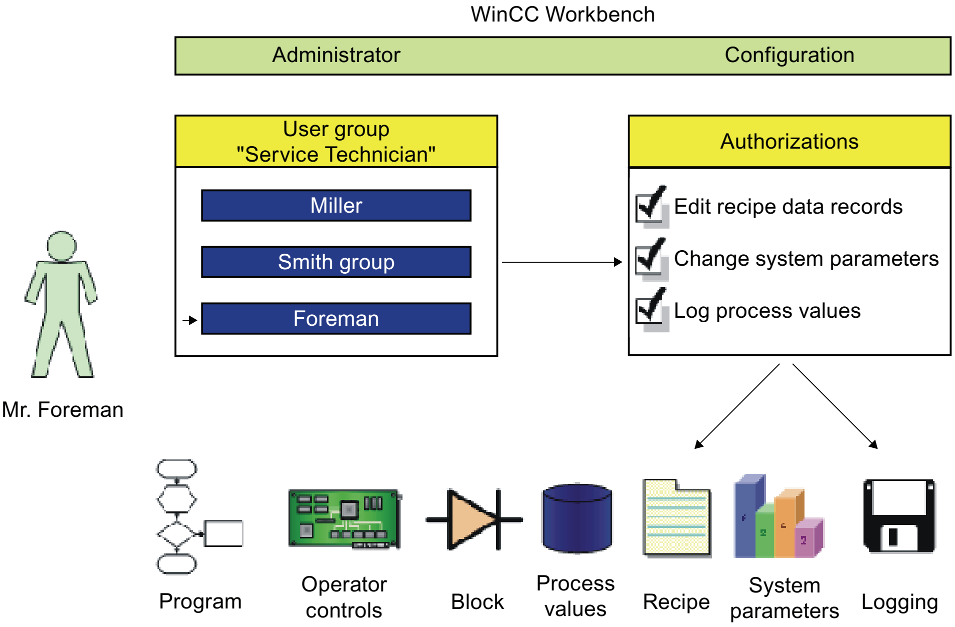
CAUTION
Access protection does not protect against incorrect operations. It is your job to ensure that only authorized personnel with appropriate training will design, commission, operate and maintain plants and machines. Access protection is not suitable for defining work routines and monitoring their observance.

10.5.2 Form of the user administration

Introduction

In case of a project in manufacturing engineering, the environment at the equipment manufacturer has to be differentiated from the environment at the end customer as plant operator.

The equipment manufacturer allows users, for example Mr. Foreman, a specific access within the application or HMI device. However, a user Foreman does not exist at the end customer. The machine manufacturer cannot know the end users and the tasks they have to perform for configuration. The final users are usually set after commissioning at the end customer.



Principle

To minimize the work required for management, authorizations are assigned via user groups and not directly to individual users.

A user group assembles configured authorizations according to common jobs. For example, all permissions required for a service job are collected in a "Service technician" group. When you create a user who should be responsible for servicing, you simply assign him to the "Service technician" group.

The user view enables user administration in Runtime. Use user view to create, delete and assign an authorization to users in Runtime.

The user administration separates the administration of the users from the configuration of the authorizations. This ensures flexibility at the access protection.

Defaults can be set for the user administration during the configuration phase in the Engineering System.

10.5.3 Basics

10.5.3.1 Users

Introduction

You can create users in the "Users" tab of the "User administration" editor and assign them to user groups. The "Users" tab is part of the user administration in WinCC.

Open

To open the "Users" tab, double-click "User administration" in the project window.

Work area

The users are managed in the work area:

- You create or delete users.
- You assign users to user groups.

Note

You can assign a user to exactly one user group.

Inspector window

When you select a user, you can change the password in the "General" group. Under "Automatic logoff" you can specify if the user is to be automatically logged off by the HMI device when there is no operator activity after the specified time.

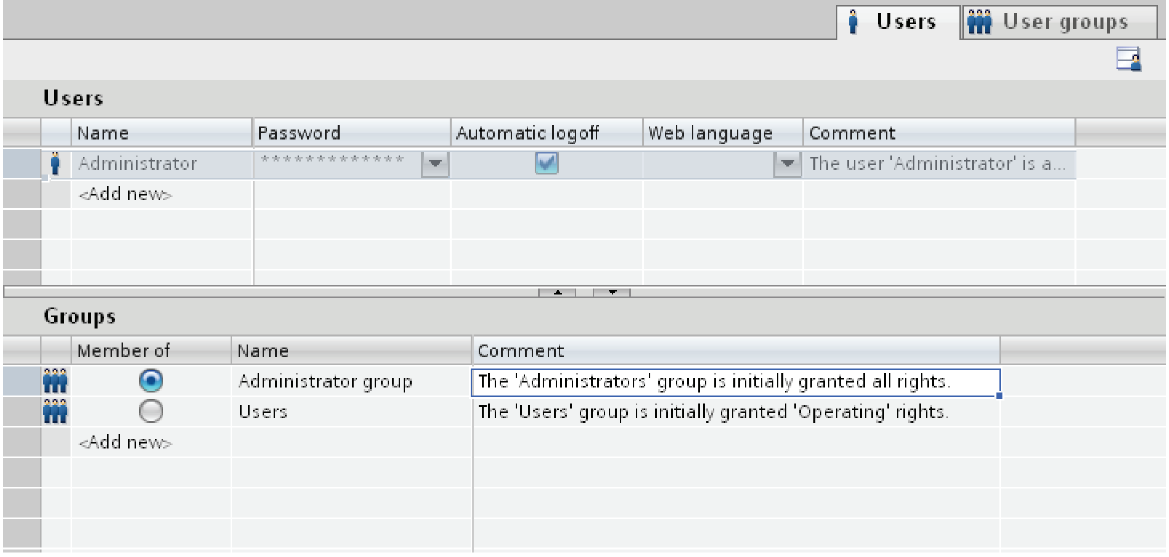
10.5.3.2 Users work area

Introduction

The "Users" work area lists the users and user groups in table form. You administrate the users and assign them to a user group.

Principle

The work area consists of the "Users" and "Groups" tables.



Users

Name	Password	Automatic logoff	Web language	Comment
Administrator	*************	☑		The user 'Administrator' is a...
<Add new>				

Groups

Member of	Name	Comment
●	Administrator group	The 'Administrators' group is initially granted all rights.
○	Users	The 'Users' group is initially granted 'Operating' rights.
<Add new>		

The "Users" table shows the existing users. When you select a user in this table, the "Groups" table shows the user group to which the user belongs.

10.5.3.3 User groups

Introduction

You can create users and authorizations in the "User groups" tab of the "User Administration" editor. The "User groups" tab is part of the user administration in WinCC.

Open

Double-click the "User administration" in the project window. Open the "User groups" tab.

Work area

The user groups and authorizations are managed in the work area:

- You create new user groups and authorizations or delete them.
- You assign the authorizations to the user groups.

Inspector window

When a user group or an authorization is selected, you can edit the name in the "General" group. You can also enter a brief description in the "Comment" group.

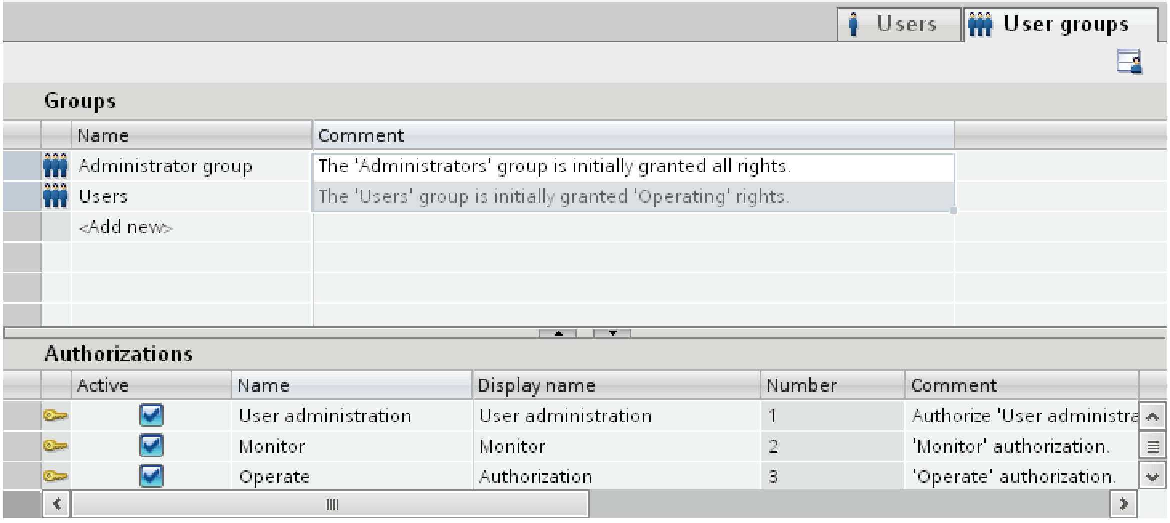
10.5.3.4 User groups work area

Introduction

The "User groups" work area shows a table of the groups and their authorizations. You administer the user groups and assign authorizations to them.

Principle

The work area consists of the "Groups" and "Authorizations" tables.



Groups

Name	Comment
Administrator group	The 'Administrators' group is initially granted all rights.
Users	The 'Users' group is initially granted 'Operating' rights.
<Add new>	

Authorizations

Active	Name	Display name	Number	Comment
☑	User administration	User administration	1	Authorize 'User administra
☑	Monitor	Monitor	2	'Monitor' authorization.
☑	Operate	Authorization	3	'Operate' authorization.

The "Groups" table shows the existing user groups. When you select a user group in this table, the "Active" column of the "Authorizations" table shows the authorizations which were assigned to the user group.

The number of the user group and of the authorization is assigned by the user administration. The designations and descriptions are assigned by you.

The number of predefined authorizations are fixed. Authorizations that you create can be freely edited. Ensure that the assigned numbers are unique.

10.5.3.5 Settings for the user administration

Introduction

In the "Runtime settings > User administration" editor, configure the security settings for users and their passwords in runtime.

Open

Double-click the "Runtime settings" editor in the project window. Click "User administration."

Work area

You carry out the settings for the validity of passwords in runtime in the work area. You determine the complexity of the password, for example.

Effects in runtime

The security settings have the following effects in runtime, depending on the configuration.

- "Runtime services" group
  - "Enable limit for logon attempts" check box selected

    The number entered in the "Number of incorrect logon attempts" box defines the number of logon attempts a user is allowed before being assigned to the "Unauthorized" group.

    "Enable limit for logon attempts" check box not selected

    The user has an unlimited number of logon attempts in runtime.
  - "Number of incorrect logon attempts" field

    If you enter "4" in the field, for example, and the fourth logon attempt fails, the user is automatically assigned to the "Unauthorized" group.

    You can specify 1 to 9 attempts.
  - "Logon only with password" check box

    When the check box is selected, the user will be authenticated by the password. The user name is not required.

    To match users to passwords, you cannot configure passwords more than once.
- "Hierarchy level" group
  - "Group-specific rights for user administration" check box

    When this check box is selected, administrators only manage users whose group number is less than or equal to their own.

    For example, an administrator whose group number is 5 can only manage users whose group number is less than or equal to 5. This means that the administrator can assign users only to groups with a group number less than or equal to 5.

- "Password" group
  - "Enable password aging" checkbox selected

    The password expires after the number of days set in the "Validity of the password (days)" field.

    The "Password aging" column is selected in the "User groups" editor. This enables you to specify group-by-group, if the passwords should expire or if the password generations should be saved. Passwords never expire for groups for which password aging is not enabled.
  - "Prewarning time (days)" field

    The user is informed that the password will expire the specified number of days before the password expires.
  - "Password generations" field

    If the user changes the password, the new password must be different from the specified number of previous passwords. The number of password generations ranges from 1 to 5.
- "Password complexity" group
  - "Must include special characters" check box selected

    The user must enter a password containing at least one special character at any position.
  - "Must include number" check box selected

    The user must enter a password containing at least one number at any position.
  - "Minimum password length" field

    The user must enter a password with a minimum length, as specified in the "Minimum password length" field.

    The minimum length of the password is 3 characters.

10.5.4 Setting up the user administration

10.5.4.1 Basics on user administration

Principle

This section addresses four target groups. The topics are organized correspondingly. The target groups serve as examples for different groups of persons who use the user administration.

1. Administrator OEM
2. Administrator RT
3. Planners
4. Operator

As Administrator OEM you create the user groups, users and authorizations for Runtime in the Engineering System of, for example, an equipment manufacturer.

As Administrator RT you administer users in Runtime by means of the "User view."

As the project engineer you assign the authorizations to the user groups in the Engineering System. In addition, you configure the authorizations for objects.

As Operator you log on in runtime. You can only access a protected object if you have the required authorization.

Note

The Administrator RT target group already exists in the Runtime user administration as the predefined user group "Administrator group." For a better understanding the predefined user groups and authorizations are not used below.

10.5.4.2 Administering users for Runtime

Creating an authorization

Introduction

You create an authorization and assign it to one or more user groups.

Requirements

The "User groups" work area is open.

Procedure

1. Double-click "Add" in the "Authorizations" table.
2. Enter "Archive data" as the name of the authorization.
3. Enter a brief description as the "Comment."

Creating a user group

Introduction

User groups are created so that you do not have to assign an authorization to every single user. You create a user group, assign authorizations and then assign users to it.

Note

The name of the user group has to be unique within the project. Otherwise the input is not accepted.

Note

Using SIMATIC Logon

Ensure that the names of the user groups in Windows and WinCC are identical.

Requirements

The "User groups" work area is open.

Procedure

1. Double-click "Add" in the "Groups" table.
2. Enter "Operators" as the "Name" of the user group.
3. Change the "Number" of the user group as required.
4. Enter "Display name" of the "Operators" user group.
5. Enter a brief description as the "Comment".

In runtime, the user view shows the display name of the user group. The display name of the user group depends on the language. You can specify the name in several languages and switch between languages in runtime.

See also

Assigning an authorization (Page 2260)

Creating users (Page 2261)

Assigning a user to a user group (Page 2262)

Managing user groups (Page 2263)

User administration (Page 2262)

Assigning an authorization

Introduction

When you allocate an authorization to a user group, all assigned users have this authorization.

Requirements

- An "Archive data" authorization has been created.
- An "Operators" user group has been created.
- The "User groups" work area is open.

Procedure

1. Click on the "Operators" user group in the "Groups" table. The "Authorizations" table shows all authorizations.
2. Activate the "Archive data" authorization in the "Authorizations" table.

NOTICE
The "Archive data" authorization is only a designation and does not having any relation to the function "Archiving." You have to establish this relation yourself. To do so, configure the "StartArchiving" system functions at a control button and select "Archive data" as the "Authorization."

See also

Creating a user group (Page 2259)

Creating users

Introduction

You create a user so that users can log on with their user names in runtime after loading to the device.

As an alternative, you can create and change users by means of the "User view" in Runtime.

In order for a created user to have authorizations you have to assign him to a user group and allocate authorizations to the user group.

The logon is successful when the user name entered during the logon matches a user in Runtime. In addition, the entered password must agree with the stored password of the user.

Note

Note that the entry is case-sensitive.

Requirements

The "Users" work area is open.

Procedure

1. Double-click "Add" in the "Users" table.
2. Enter "Foreman" as the user name.

   Note

   The user name must be unique within the project. Otherwise the input is not accepted.

3. Click the ▾ button in the "Password" column. A dialog box for entering the password is displayed.
4. Enter the password of the user.
5. To confirm the password enter it a second time in the lower field.
6. Close the dialog box by using the ✓ icon.
7. If a user is to be logged off after a specific time period, activate "Automatic logoff".
8. Click in the "Logoff time" column. The preset value for "Logoff time" is 5 minutes.
9. Enter a brief description as the "Comment".

See also

Creating a user group (Page 2259)

Assigning a user to a user group

Introduction

When you assign a user to a user group, the user has the authorizations of the user group.

Note

You have to assign a user to exactly one user group. The assignment is checked during the consistency check and generation of the project.

Requirements

- The user "Foreman" has been created.
- An "Operators" user group has been created.
- The "Users" work area is open.

Procedure

1. Click on the "Foreman" user in the "Users" table. The "Groups" table shows all user groups.
2. Activate the "Operators" user group in the "Groups" table.

See also

Creating a user group (Page 2259)

User administration

Introduction

In the work area, you can administer users and assign them to user groups.

Requirements

The "Users" work area is open.

Changing the user name

1. In the "Users" table, double-click the field in the "Name" column to change the user name.
2. Change the user name.
3. Confirm your entry with <Return>.

Alternatively, select the user in the work area. Change the user name under "Properties > Properties > General" in the Inspector window.

Changing the password of the user

1. Click the ▼ button in the "Password" column of the "Users" table. A dialog for entering the password opens.
2. In the "Enter password" field, enter the new password.
3. Reenter the new password in the "Confirm password" field.
4. Confirm your entry with <Return>.

Alternatively, select the user in the work area. Change the password under "Properties > Properties > General" in the Inspector window.

Displaying invisible columns

1. Position the mouse cursor on the title of the "Users" table.
2. Right-click to open the shortcut menu and enable the display of the "Logoff time" column, for example.

Changing the logoff time of the user

1. In the "Users" area, double-click on the field in the "Logoff time" column to change the logoff time.
2. Change the logoff time.
3. Confirm your entry with <Return>.

Alternatively, select the user in the work area. Change the logoff time under "Properties > Properties > Automatic logoff" in the Inspector window.

Deleting a user

1. Select the line of the user to be deleted.
2. Open the shortcut menu with the right mouse button and select the "Delete" command.

> **Note**
>
> Predefined users cannot be deleted.

See also

Creating a user group (Page 2259)

Managing user groups

Introduction

In the workplace you administer user groups and assign authorizations for use in runtime.

Requirements

The "User groups" work area is open.

Changing the name of the user group

1. In the "Groups" table, double-click the field in the "Name" column to change the name of the user group.
2. Change the name of the user group.
3. Confirm your entry with <Return>.

Alternatively, select the user group in the work area. Change the name under "Properties > Properties > General" in the Inspector window.

Note

Predefined user groups cannot be deleted.

Displaying invisible columns

1. Position the mouse cursor on the title of the "Users" table.
2. Right-click to open the shortcut menu and enable the display of the "Display name" column, for example.

Changing the displayed name of the user group

1. In the "Groups" table, double-click the field in the "Display name" column to change the display name of the user group.
2. Change the displayed name of the user group.
3. Confirm your entry with <Return>.

Alternatively, select the user group in the work area. Change the display name under "Properties > Properties > General" in the Inspector window.

Deleting a user group

1. Mark the line of the user group to be deleted.
2. Open the shortcut menu with the right mouse button and select the "Delete" command.

Note

Predefined user groups cannot be deleted.

Changing the name of the authorization

1. In the "Authorizations" table, double-click the field in the "Name" column to change the name of the authorization.
2. Change the name of the authorization.
3. Confirm your entry with <Return>.

Alternatively, select the authorization in the work area. Change the name under "Properties > Properties > General" in the Inspector window.

Deleting authorizations

1. Mark the line of the authorization to be deleted.
2. Open the shortcut menu with the right mouse button and select the "Delete" command.

Note

Predefined authorizations cannot be deleted.

See also

Creating a user group (Page 2259)

10.5.4.3 Managing users in Runtime

Users in Runtime

Principle

You create users and user groups and assign authorizations to them. You configure objects with authorizations. After download to the HMI device, all objects which were configured with an authorization are protected against unauthorized access in Runtime.

User view

When you configure a user view in the Engineering System, you administer users in this user view following download to the HMI device.

CAUTION
Changes in the user view are effective immediately in Runtime. Changes in runtime are not updated in the engineering system. When downloading the user administration to the HMI device, all changes in the user view are overwritten after a security prompt and based on the settings.

Users who have a "User administration" authorization have unlimited access to the user view. This allows them to administer all users. Any other user has only limited access to the user view for self administration.

User view

Purpose

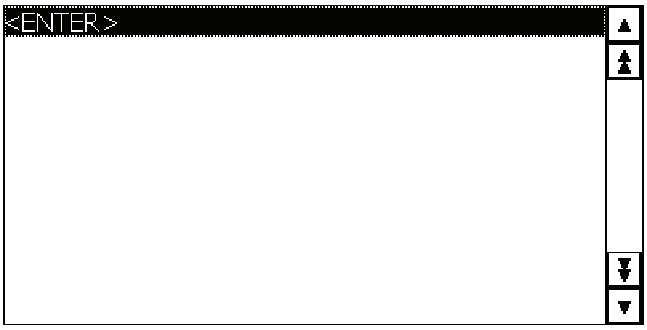
You configure a user view in the engineering system to also administer the users in Runtime.

Structure

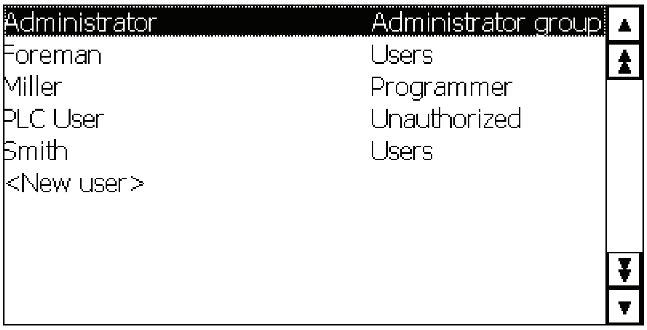
The user view shows the following in each line:

- The user
- The corresponding user group.

If no user is logged on, the user view is empty. The content of the individual fields is displayed after logon.



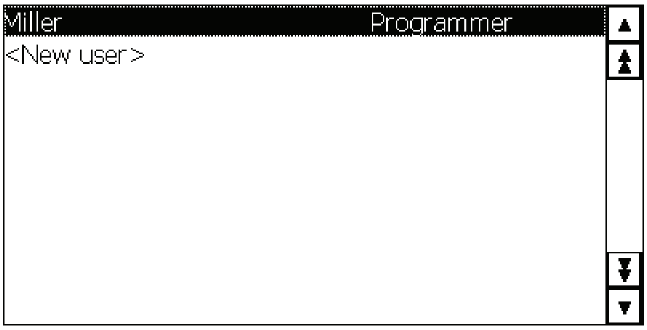
User view of an administrator



Administrator	Administrator group
Foreman	Users
Miller	Programmer
PLC User	Unauthorized
Smith	Users
<New user>	

When an administrator is logged on, the user view shows all the users. The administrator changes the user name and the password. The administrator creates new users and assigns them to an existing user group.

User view of a user



Miller	Programmer
<New user>	

When no administrator is logged on, the user view shows only the logged-on user. Users can change their own passwords.

Configuring a user view

Introduction

You configure a user view in the Engineering System to also administer users in Runtime.

Requirements

A screen has been created.

Procedure

1. Select the "User view" object from the "Controls" category in the toolbox.
2. Drag-and-drop the "User view" object into the screen.
3. Click on "Properties > Properties" in the Inspector window.
4. Specify the appearance of the "User view".
5. You can, for example, select " "Display mode > Fit to size > Fit object to contents".

Result

You have created a user view in the screen.

Creating users

Introduction

You create a user so that users can log on under their user name in runtime.

As an alternative, you can create users in the engineering system and download them to the HMI device.

The logon is successful only when the user name entered during the logon matches a user in runtime. In addition, the password entered at logon has to match that of the user.

Note

Note that the entry is case-sensitive.

You assign the user to a user group. The user then has the authorizations of the user group.

NOTICE

Runtime users must be assigned to a user group. The user group is created in the Engineering System. The designation of the user group is language-dependent.

Requirements

- The user view is open.
- A "Group 2" user group has been created.

Procedure

1. Click "<New User>" in the user view. A dialog box opens.
2. Enter "Foreman" as the user name.
3. Press the <Return> button.
4. Click "Password."
5. Enter the password of the user.
6. Press the <Return> button. The password is hidden.
7. Click in the "Group" column.
8. Select "Group 2" as the "Group".
9. Press the <Return> button.
10. Click in the "Logoff time" column.
11. Enter the time after which the user is logged off automatically.

Managing users in the simple user view

Introduction

If you have configured a user view in the engineering system, the users and user groups can be managed in runtime.

CAUTION
Changes in the user view are effective immediately in runtime. Changes in runtime are not updated in the Engineering System. When downloading the user administration to the HMI device, all changes in the user view are overwritten after a security prompt and based on the settings.

Requirements

- Runtime is enabled.
- The simple user view has been created.
- The screen with the simple user view is open.
- You have the default "User administration" authorization.

NOTICE
If you do not have a "User administration" authorization, you can only change your own password and your logoff time.

Changing a user name

1. Click on the line of the user whose name you want to change. A dialog box opens.
2. Enter a new user name.
3. Click "OK" to confirm your entry.

NOTICE
The user can then no longer log on with his previous password in runtime. If you delete the name and press <Return>, the user is deleted.

Changing the user password in basic user display

1. Click on the line of the user whose password you want to change. A dialog box opens.
2. Enter a new password.
3. Click "OK" to confirm your entry.

NOTICE
The user can then no longer log on with his previous password in runtime. If you delete the password in the basic user view and press <Return>, a message will be generated. The message specifies that the password does not lie within the defined limits.

Changing the logoff time of the user

1. Click on the line of the user whose logoff time you want to change. A dialog box opens.
2. Enter a new logoff time.
3. Click "OK" to confirm your entry.

Deleting a user

1. Click the name of the user to be deleted.
2. Delete the name.
3. Press the <Return> button.

NOTICE
The user can no longer log on in runtime.

Assigning a user to a different user group

1. Click on the line of the user whose user group you want to change. A dialog box opens.
2. Click on the "User group" box.
3. Select a user group.
4. Click "OK" to confirm your selection.

Unlocking users

Unlock locked out users

The check box "Activate limit for login attempts" is activated in the "Runtime settings > User administration".
The number 3 is entered in the field "Number of invalid login attempts".

If users have three failed attempts at login, e.g. by entering an incorrect password, they are assigned to the "Unauthorized" group. The user loses all authorizations. The user can still log on, but no longer has any authorizations. Only a user with administrator rights re-assigns the unauthorized user to a user group.

Logging on as a user

Introduction

As a rule you log-on as a user by means of a special button. The logon dialog box is displayed to this purpose.

The logon dialog box is displayed by default during access to a protected object if

- No user is logged on in Runtime.
- The logged-on user does not have the required authorization.

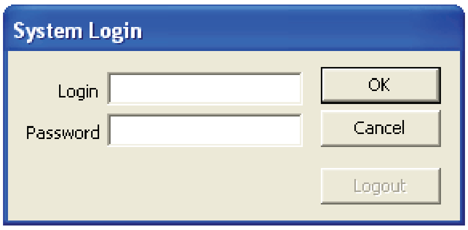
> **Note**
>
> The system always opens the logon dialog on the OP 73, OP 77A, TP 177A and Basic Panels HMI devices when you press an access-protected button:

Requirements

- Under "Runtime settings > User administration" the
  - "Enable limit for logon attempts" check box has been selected.
  - The number 3 is entered in the field "Number of invalid login attempts".
- The "ShowLogonDialog" system function is configured on a button called "Logon".

Procedure

1. Click the "Logon" button. The logon dialog box is displayed.



2. Enter your user name as it was specified in the user administration, for example "Foreman".

   If someone has logged on before you, the name of the user will be displayed.
3. Enter the corresponding password. The input is concealed.
4. Click "OK" to close the dialog box.

Logon was successful

If you have entered the user name "Foreman" and the entered password corresponds with the stored password, you are logged on as the user "Foreman" in Runtime. You have the authorizations of the user "Foreman".

When the user "Foreman" accesses a protected object such as the "Logging" button, access to this protected object will only be authorized if the user "Foreman" has the required authorization. The programmed function is executed immediately.

If you do not have the required authorization after the successful log-on, a corresponding error message is displayed. However, you remain logged on in Runtime.

Logon was unsuccessful

An error message is displayed.

In order to maintain security, you or the user logged-on before you no longer has any authorizations. However, access to unprotected objects remains possible. The user view does not, however, show any entries. The user view and the authorizations change after the next successful log-on.

If the third login attempt has failed, the user will be assigned to the "Unauthorized" group. For this reason, do not configure a user group with this display name.

If the "Log off" function is called up or the logoff time of the user has expired, the user is logged off.

10.5.4.4 Configuring access protection

Access protection

Introduction

You configure an authorization at an object in order to protect it against access. All logged-on users who have this authorization can then access the object. If a user does not have authorization to operate an object, the logon dialog is displayed automatically.

Note

Several system functions are available under "User administration" so that user, password, and user group can be edited, for example, in the PLC.

Configuring authorization (Basic,Advanced; Professional)

Introduction

You configure the "Archive data" authorization for a button. Then only those users who have the appropriate authorization have access to this button, for example all the users of the "Operators" user group.

This ensures that access to the command button is protected. If a logged-on user who belongs to the "Operators" user group and has the required authorizations clicks the button, alarms and variables are archived.

An example describes in detail how to configure a command button with access protection.

Requirements

- The "Operators" user group has been created.
- The "Archive data" authorization has been created.
- A screen has been created and opened.
- The screen contains a button.

Procedure

1. Click the button in the screen.
2. Click "Properties > Properties > Security" in the Inspector window.
3. Select "Archive data" as the "Authorization".
4. In the Inspector window, select "Properties > Events > Click".
5. Select a system function from the function list.

Note

The "Enable" and "Disable" events are only used to detect whether an object was selected or deselected. The events do not, however, trigger a password prompt.

Do not use the "Enable"or "Disable" event if you want to configure access protection at the function call of the object.

10.5.5 Reference

10.5.5.1 Objects with access protection

Introduction

The following objects can be configured with an authorization:

- Date/time field
- I/O field
- Graphic I/O field
- Recipe view
- Switch
- Button
- Symbolic I/O field

10.5.5.2 Default user groups and authorizations

Principle

The predefined user groups and authorizations have the following numbers:

User group	Number
"Administrator group"	1
"Users"	2

Authorization	Number
"User administration"	1
"Monitor"	2
"Operate"	3

10.5.6 Examples

10.5.6.1 Example: Configuring a button with logon dialog box

Task

In the following example, you configure the function "ShowLogonDialog" for a button. A different user can then log on in runtime when the shift changes, for example. In the process the user previously logged on is logged off.

Note

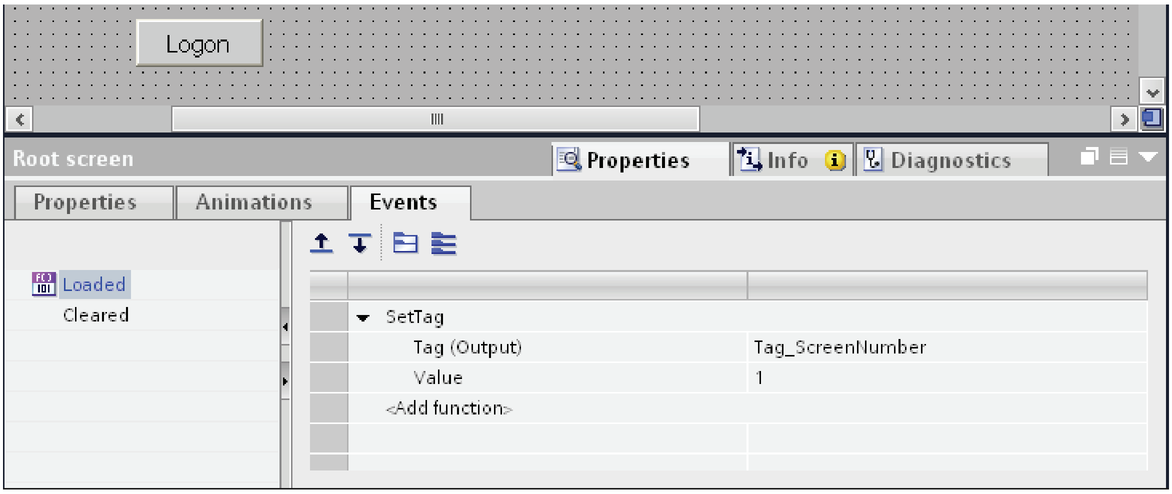
In runtime the logon dialog box is not displayed by default until you access a protected object. Either no user is logged on or the logged-on user does not have the required authorization.

Requirements

- A screen has been created.
- A button has been created in the screen.

Procedure

1. Click the button in the screen.
2. Click "Properties > Events > Release" in the Inspector window.
3. Click the entry "Add function" in the "Function list" table.
4. Select the system function "ShowLogonDialog" from the "User administration" group.



SetTag	
Tag (Output)	Tag_ScreenNumber
Value	1
<Add function>	

Result

If the user clicks on the button in runtime, the function "ShowLogonDialog" is called up. When the function "ShowLogonDialog" is called up, the logon dialog box is displayed. The user logs on with his user name and password.

10.5.6.2 Example: Logging the logon and logoff events

Task

In the following example, you configure the function "TraceUserChange" to the event "User change".

Principle

The "TraceUserChange" function is called when a user logs on or off. When a function is called up, a system message with information about the corresponding user is output.

This system message can be archived. When archiving, the system message is provided with a date stamp and time stamp. This ensures that you can track which user was logged on at the HMI device at which time and for how long.

Requirements

- You have created an HMI device with Runtime Advanced.
- The Inspector window is open.

Procedure

1. Double-click the "Scheduler" in the Project view.
2. Double-click "Add" in the table of the tasks.
3. Enter "Logon-Protocol" as the "Name".
4. Select "User change" as the "Trigger".
5. Open "Properties > Events" in the Inspector window.
6. Click the entry "Add function" in the "Function list" table.
7. Select the "TraceUserChange" system function.

Result

A system message is output when a user logs on or logs off.

10.5.6.3 Example of user management

Example: Structure of user management

Task

In the following example you set up a user administration for different users and user groups. The example orientates itself to a typical requirement profile from manufacturing engineering.

Principle

Completely different groups of persons are involved in a plant or project. Each group of persons protects its respective data and functions against access by others. For this purpose, users are created and assigned to a user group.

You can reproduce different views through user groups.

Example:

- Organizational view: Commissioners, Operators, Shift I, Shift II
- Technological view: Axis control, Tool changers, Plant North, Plant South

The following example orientates itself to the organizational view.

Every user group has characteristic requirements regarding access protection: A user group has operating authorizations for specific application cases. A programmer changes, for example, recipe data records.

In the example the users Miller, Group Smith and Foreman are created and assigned to different user groups.

Ms. Miller works as a programmer in the engineering system. The Group Smith are commissioners. Mr. Foreman is an operator.

Requirements

- A new project has been created.
- The "User administration" editor is open.

Procedures overview

Working with user administration has the following procedure in the example:

1. Creating authorizations The planner specifies which authorizations are required for access protection.
2. Configuring authorizations: The planner specifies which objects may be operated and which functions may be executed.
3. Creating user groups and allocating authorizations: The administrator creates the user groups together with the planner. The planner uses the authorizations to specify who may operate objects and change parameters.
4. Creating users and assigning them to a user group: The administrator administers the users.

Result

The aim is the following structure of the user administration of users, user groups and authorizations:

Users			User groups	Authorizations			
Miller	Smith	Foreman	Roles	Changing recipe records	Changing system parameters	Changing process parameters	Managing
			Administrator group				x
X			Programmer	X			
	X		Commissioning engineers	X	X	X	
		X	Operators	x			

The user "Foreman" who belongs to the "Operators" user group has access to the configured "To Recipe view" button.

Note

Alternatively, you can create several users as operators with different operating authorizations, for example, Operator Level 1, Operator Level 2.

Example: Creating and configuring authorizations

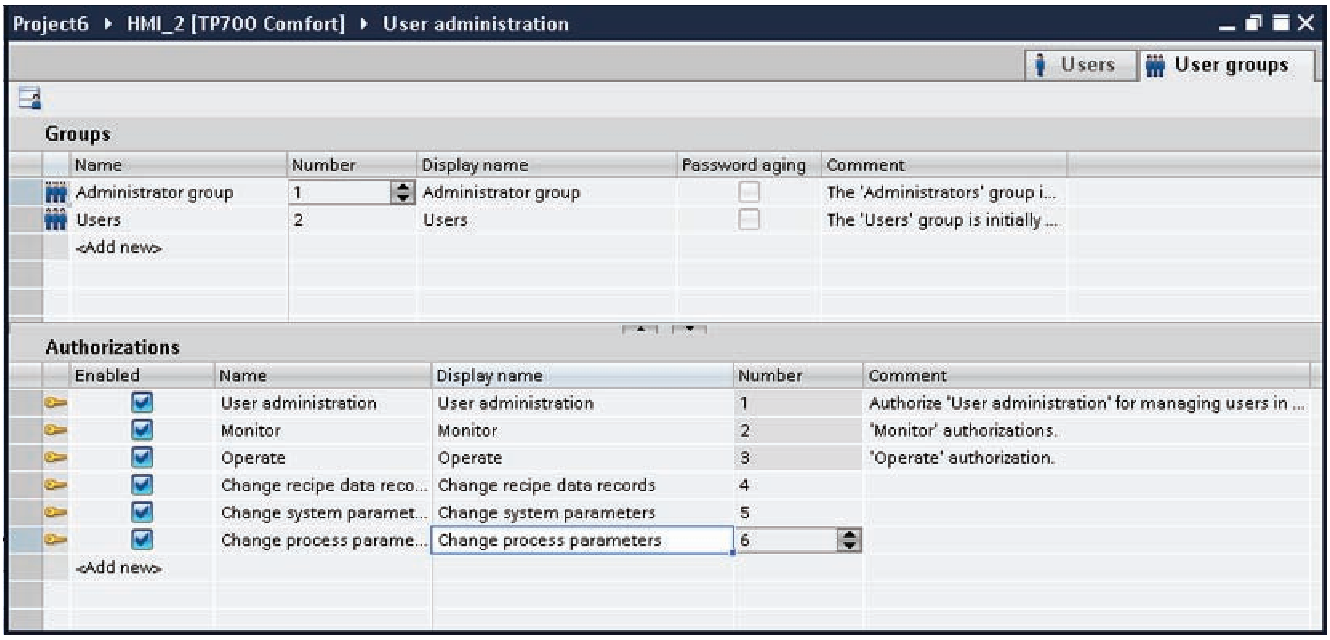
Task

The following example shows you how to create the authorizations

Procedure

1. Open the "User groups" work area.
2. Double-click "Add" in the "Authorizations" table.
3. Enter "Change recipe data records" as the "Name" of the authorization.
4. Repeat steps 2 and 3 to create additional authorizations: "Change system parameters", "Change process parameters".

Result



Groups

Name	Number	Display name	Password aging	Comment
Administrator group	1	Administrator group	□	The 'Administrators' group i...
Users	2	Users	□	The 'Users' group is initially ...
<Add new>				

Authorizations

Enabled	Name	Display name	Number	Comment
☑	User administration	User administration	1	Authorize 'User administration' for managing users in ...
☑	Monitor	Monitor	2	'Monitor' authorizations.
☑	Operate	Operate	3	'Operate' authorization.
☑	Change recipe data reco...	Change recipe data records	4	
☑	Change system paramet...	Change system parameters	5	
☑	Change process parame...	Change process parameters	6	
<Add new>				

Example: Configuring a button with access protection

Task

In the following example you use a system function to create a button for a screen change. You protect the "To Recipe view" button against unauthorized operation. To do so, you configure the "Change recipe data records" authorization at the "To Recipe view" button.

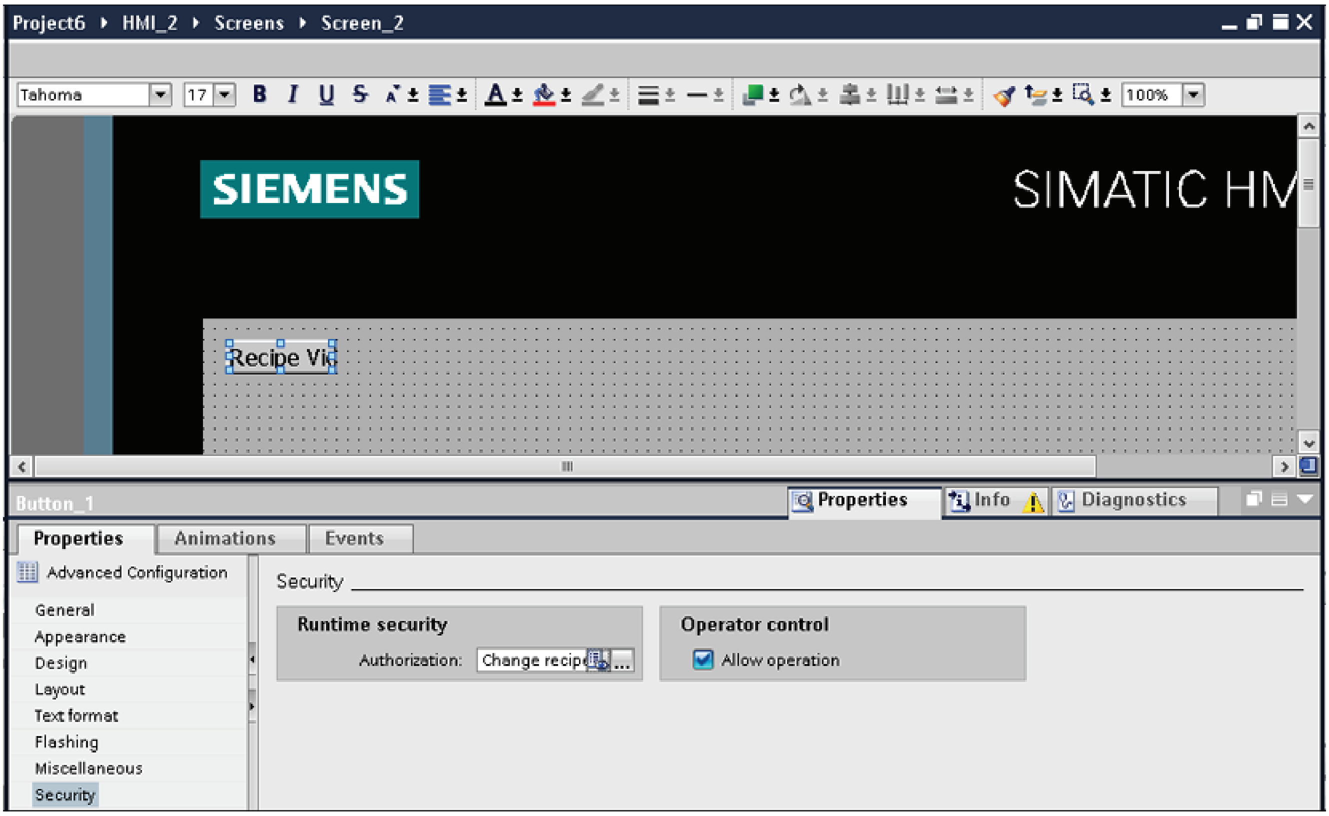
Requirements

- A "Change recipe data records" authorization has been created.
- A "Recipes" screen has been created.
- A "Start" screen has been created and opened.
- A button has been created and marked in the "Start" screen.

Procedure

1. Click "Properties > Properties > General" in the Inspector window.
2. Enter "To Recipe view" as the text.
3. Click "Properties > Events > Click" in the Inspector window.
4. Click the "Add function" entry in the first line of the "Function list" table.
5. Select the "ActivateScreen" system function in the "Screens" group.
6. Click the ... button in the "Screen name" field. A dialog box for selecting the screen opens.
7. Select the "Recipes" screen and use the ✓ button to close the dialog box.
8. Click "Properties > Properties > Security" in the Inspector window.
9. Select "Change recipe data records" as the "Authorization."

Result



Security

Runtime security
Authorization: Change recip...

Operator control
☑ Allow operation

Access to the "To Recipe view" button is protected. If, for example, the user "Smith" clicks the button in Runtime, the function "Recipe view" screen is called up. Prerequisite is that the user "Smith" has logged on correctly and has the required authorization. The "Recipes" screen contains a recipe view and other screen objects.

If the logged-on user does not have the required authorization or if no user is logged on, the "Logon dialog box" is displayed.

Example: Creating user groups and assigning authorizations:

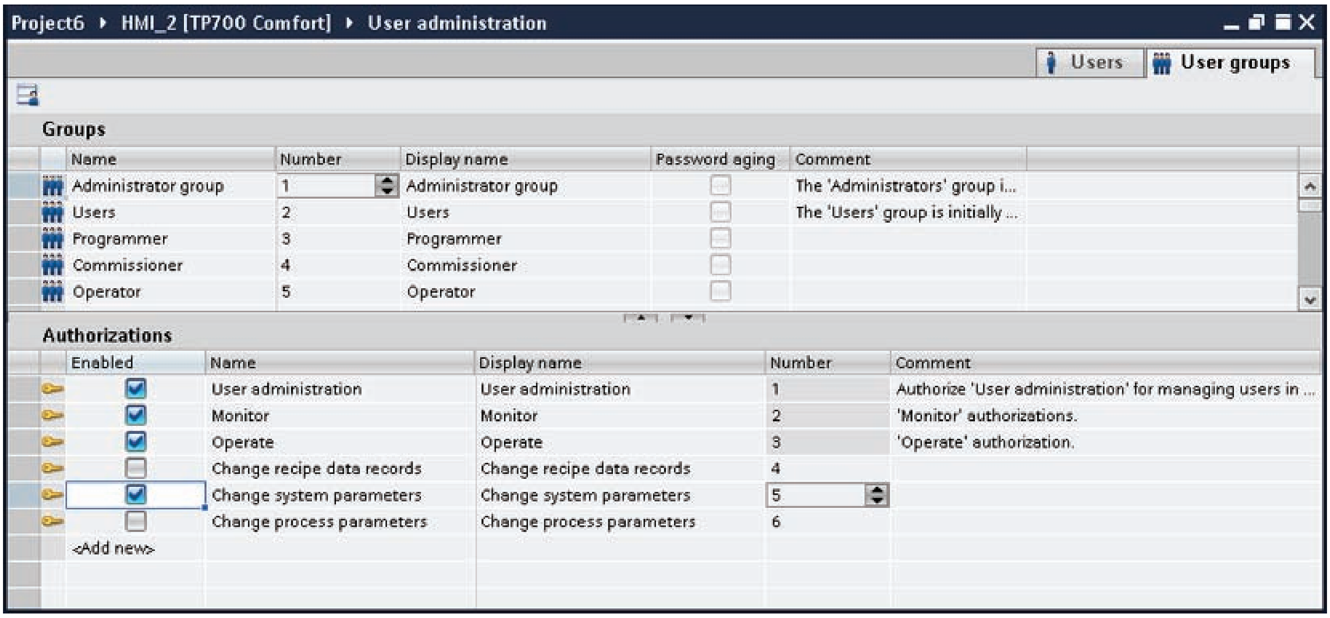
Task

In the following example you create the user groups and assign authorizations to them.

Procedure

1. Open the "User groups" work area.
2. Double-click "Add" in the "Groups" table.
3. Enter "Programmer" as the "Name".
4. Repeat steps 2 and 3 to create the "Commissioner" and "Operator" user groups.
5. Click "Administrator" in the "Groups" table.
6. Activate the "Change system parameters" authorization in the "Authorizations" table.

Interim result



Groups

Name	Number	Display name	Password aging	Comment
Administrator group	1	Administrator group	□	The 'Administrators' group i...
Users	2	Users	□	The 'Users' group is initially ...
Programmer	3	Programmer	□	
Commissioner	4	Commissioner	□	
Operator	5	Operator	□	

Authorizations

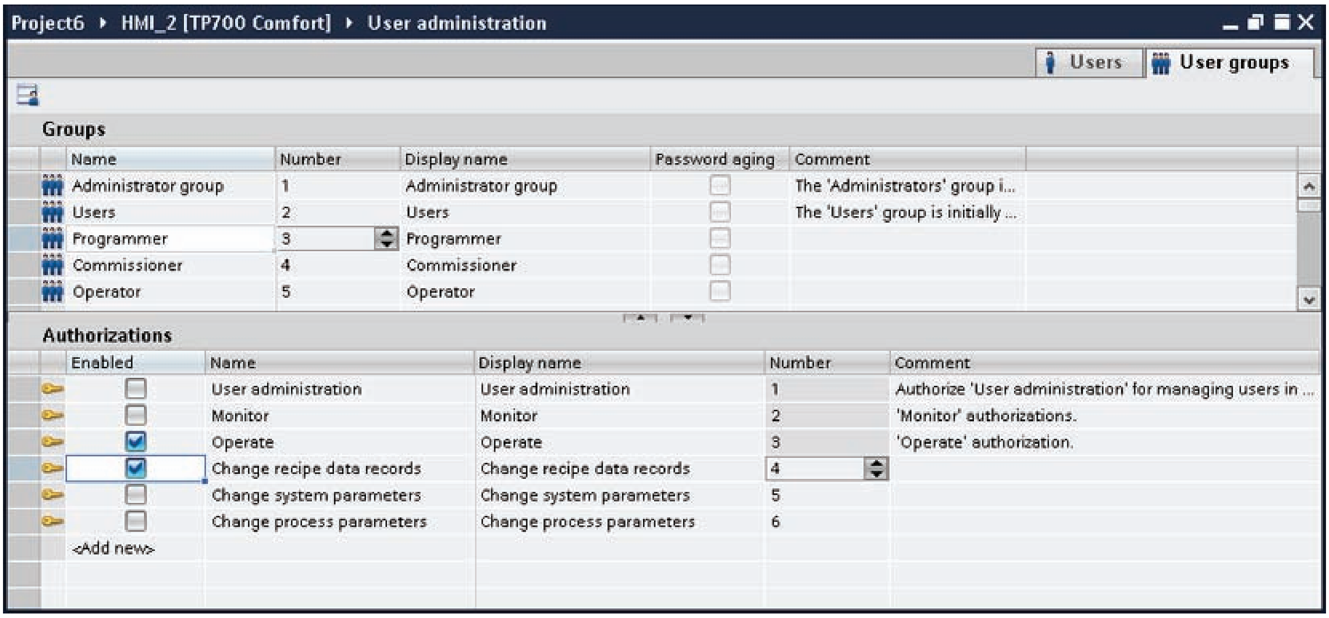
Enabled	Name	Display name	Number	Comment
☑	User administration	User administration	1	Authorize 'User administration' for managing users in ...
☑	Monitor	Monitor	2	'Monitor' authorizations.
☑	Operate	Operate	3	'Operate' authorization.
□	Change recipe data records	Change recipe data records	4	
☑	Change system parameters	Change system parameters	5	
□	Change process parameters	Change process parameters	6	
<Add new>				

Procedure

1. Click "Operator" in the "Groups" table.
2. Activate the "Change recipe data records" authorization in the "Authorizations" table.
3. Click "Commissioner" in the "Groups" table.
4. Activate the authorizations "Change recipe data records", "Change system parameters" and "Change process parameters" in the "Authorizations" table.

5. Click "Programmer" in the "Groups" table.
6. Activate the "Change recipe data records" authorization in the "Authorizations" table.

Result



Groups

Name	Number	Display name	Password aging	Comment
Administrator group	1	Administrator group	□	The 'Administrators' group i...
Users	2	Users	□	The 'Users' group is initially ...
Programmer	3	Programmer	□	
Commissioner	4	Commissioner	□	
Operator	5	Operator	□	

Authorizations

Enabled	Name	Display name	Number	Comment
□	User administration	User administration	1	Authorize 'User administration' for managing users in ...
□	Monitor	Monitor	2	'Monitor' authorizations.
☑	Operate	Operate	3	'Operate' authorization.
☑	Change recipe data records	Change recipe data records	4	
□	Change system parameters	Change system parameters	5	
□	Change process parameters	Change process parameters	6	
<Add new>				

Example: Creating users and assigning them to a user group

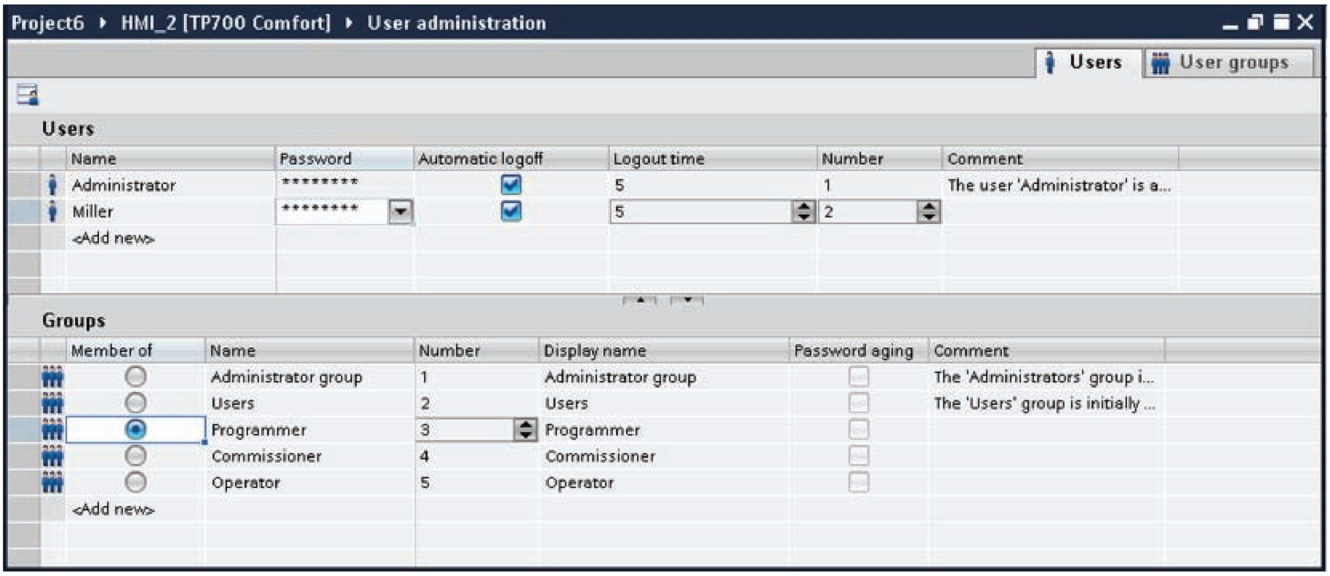
Task

In the following example you create the users and assign user groups to them. The users are sorted alphabetically immediately after the name has been entered.

Procedure

1. Open the "Users" work area.
2. Double-click "Add" in the "Users" table.
3. Enter "Miller" as the user name.
4. Click the ▾ button in the "Password" column. The dialog box for entering the password is displayed.
5. Enter "miller" as the password.
6. To confirm the password enter it a second time in the lower field.
7. Close the dialog box by using the ✓ icon.
8. Activate the "Programmer" user group in the "Groups" table.

Interim result



Users

Name	Password	Automatic logoff	Logout time	Number	Comment
Administrator	********	☑	5	1	The user 'Administrator' is a...
Miller	********	☑	5	2	
<Add new>					

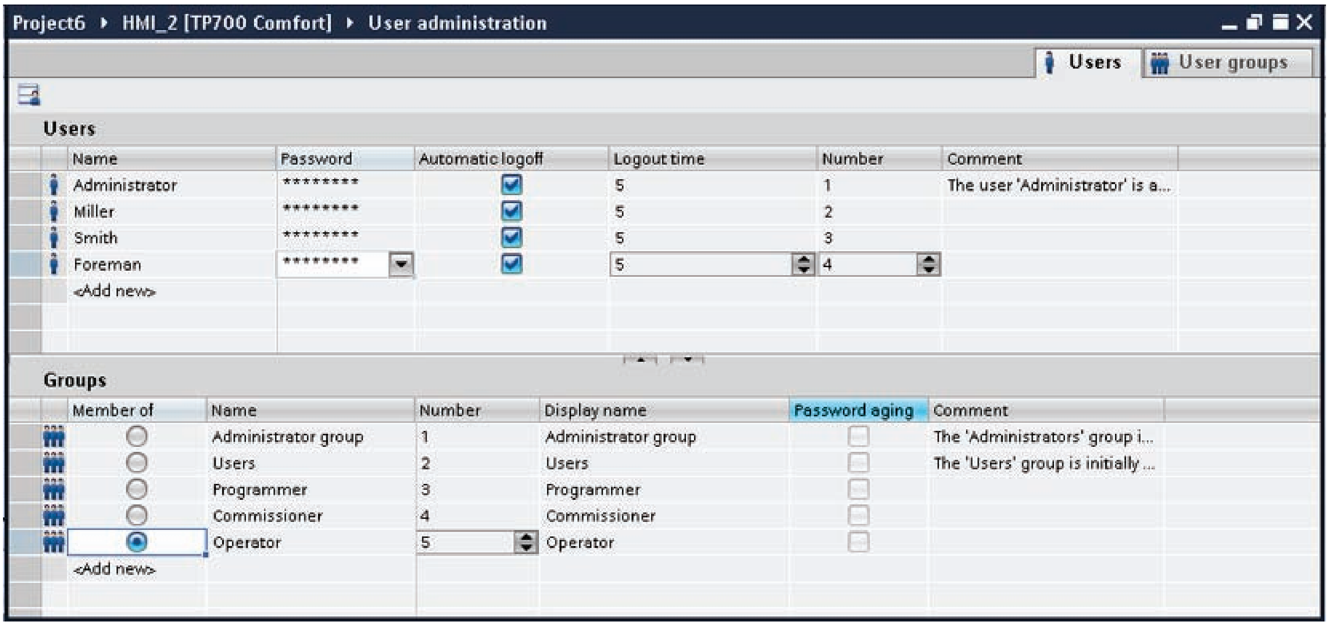
Groups

Member of	Name	Number	Display name	Password aging	Comment
○	Administrator group	1	Administrator group	☐	The 'Administrators' group i...
○	Users	2	Users	☐	The 'Users' group is initially ...
●	Programmer	3	Programmer	☐	
○	Commissioner	4	Commissioner	☐	
○	Operator	5	Operator	☐	
<Add new>					

Procedure

1. Double-click "Add" in the "Users" table.
2. Enter "Smith" as the user name.
3. Click the ▾ button in the "Password" column. The dialog box for entering the password is displayed.
4. Enter "smith" as the password.
5. To confirm the password enter it a second time in the lower field.
6. Close the dialog box by using the ✓ icon.
7. Activate the "Commissioner" user group in the "Groups" table.
8. Repeat steps 2 to 6 for the user "Foreman."
9. Activate the "Operators" user group in the "Groups" table.

Result



Users

Name	Password	Automatic logoff	Logout time	Number	Comment
Administrator	********	☑	5	1	The user 'Administrator' is a...
Miller	********	☑	5	2	
Smith	********	☑	5	3	
Foreman	********	☑	5	4	
<Add new>					

Groups

Member of	Name	Number	Display name	Password aging	Comment
○	Administrator group	1	Administrator group	□	The 'Administrators' group i...
○	Users	2	Users	□	The 'Users' group is initially ...
○	Programmer	3	Programmer	□	
○	Commissioner	4	Commissioner	□	
●	Operator	5	Operator	□	
<Add new>					

10.6 Working with system functions

10.6.1 Basics

10.6.1.1 System functions

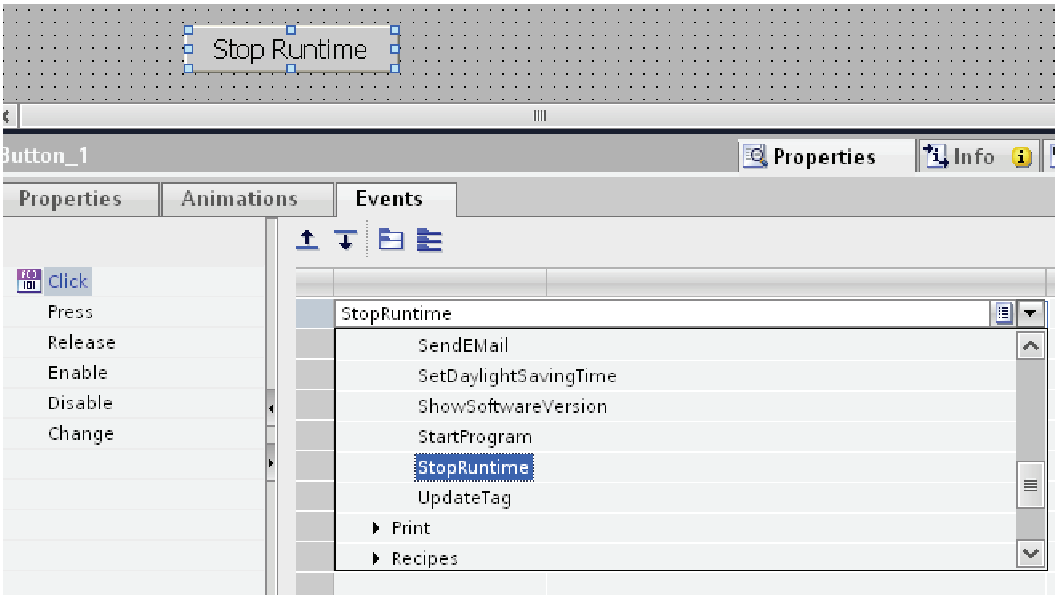
Introduction

System functions are functions supplied by WinCC. They are predefined and cannot be changed. You can use system functions to implement many tasks in Runtime even without having any programming knowledge, for example:

- Calculations, e.g. increasing a tag value by a specific or variable amount.
- Logging functions, e.g. starting a process value log.
- Settings, e.g. changing the PLC or setting a bit in the PLC.
- Alarms, e.g after a different user logs on.

Usage

You use system functions in a function list. When configuring a function list, you select the system functions from a selection list that is sorted by categories:



Each system function in WinCC is logically linked with an object and an event. As soon as the event occurs, the system function is triggered.

Language dependency

The names of the system functions are dependent on the set project language. The functionality can then be recognized immediately by the project planner.

Availability

You only configure system functions within a function list which are supported by the selected HMI device. If you use a project for several HMI devices, the system functions that are not supported by an HMI device are marked in color.

Events

The object and the selected function determine which events can be defined as triggers for executing a system function.

For example, the "Change value", "Low limit violated" and "Upper limit exceeded" events are associated with the "Tag" object. The "Loaded" and "Cleared" events are associated with the "Screen" object.

10.6.1.2 Use of system functions

Introduction

A function list is processed when a configured event occurs in runtime. The operator can trigger an event, for example, by pressing a function key on the HMI device. The system could also trigger an event if a process value falls below a specified limit, for example.

Applications

You can configure system functions on all the objects that are able to react to an event. You can use system functions directly in function lists and thereby control the sequence.

- Function list

  The system functions will be processed one line at a time in a function list. To avoid wait times, system functions in WinCC Runtime with a longer running time are executed simultaneously. For instance, a subsequent system function can already be performed even though the previous system function has not yet been completed.

An example for the configuration of a function list can be found under "Changing the operating mode on the HMI device with the current display".

10.6.2 Working with function lists

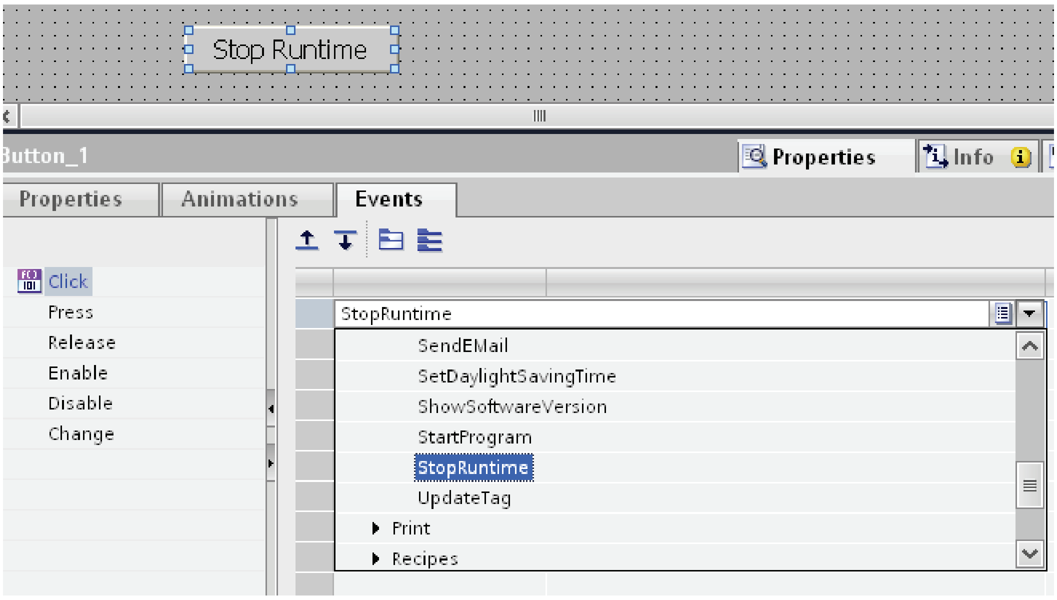
10.6.2.1 Basic of the functions list

Introduction

When the configured event occurs, several system functions can be performed with the function list.

Principle

The function list is configured for an event of an object, e.g. a screen object or a tag. The events which are available depend on the selected object and the HMI device.



Events occur only when the project is in Runtime. Events include:

- Value changes of a tag
- Pressing of a button
- Activation of Runtime

You can configure exactly one function list for each event.

Note

The choice of configurable system functions in a function list depends on the HMI device chosen.

10.6.2.2 Properties of a function list

Status information

During configuration the project data is tested in the background.

With the following causes the function list is not executed in Runtime and the incorrect entries are marked red:

- At least one system function is not completely supplied with parameters.
- At least one system function is contained which is not supported by the selected HMI device, for example, by changing the device type.

Executing system functions

System functions in a function list are executed in runtime sequentially from top to bottom. To avoid wait times, system functions with a longer running time (such as file operations) are processed simultaneously. For instance, a subsequent system function can already be performed even though the previous system function has not yet been completed.

Use a script with loops, conditional statements and abort conditions to program non-sequential and conditional procedures.

Note

Availability for specific devices

User-defined functions are not available on Basic Panels.

10.6.2.3 Configuring a function list

Introduction

You can configure a function list by selecting system functions from a drop-down list. The system functions are arranged in the drop-down list according to categories.

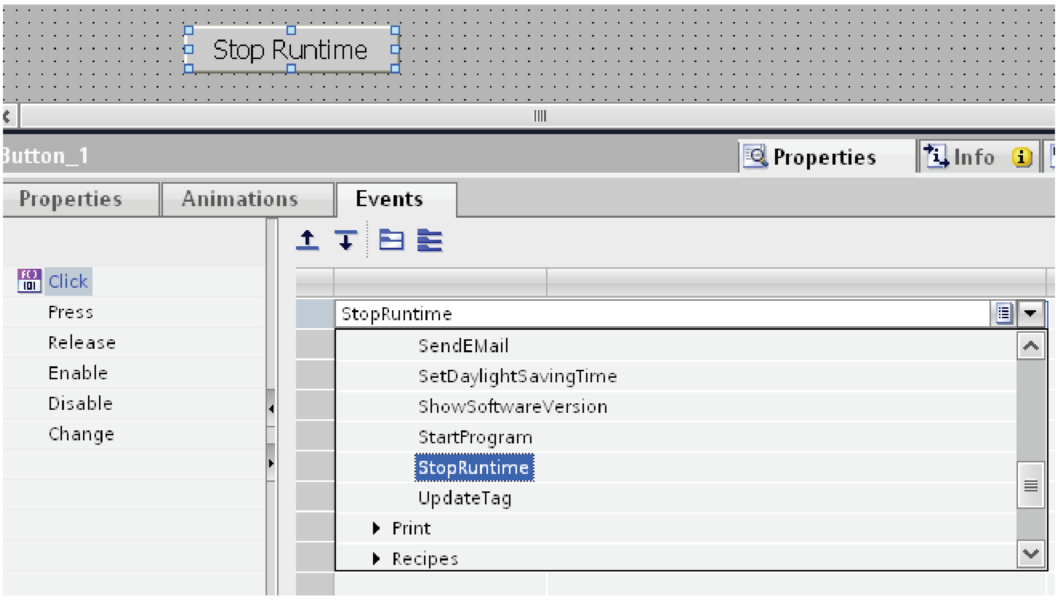
Requirement

Object has at least one configurable event.

Procedure

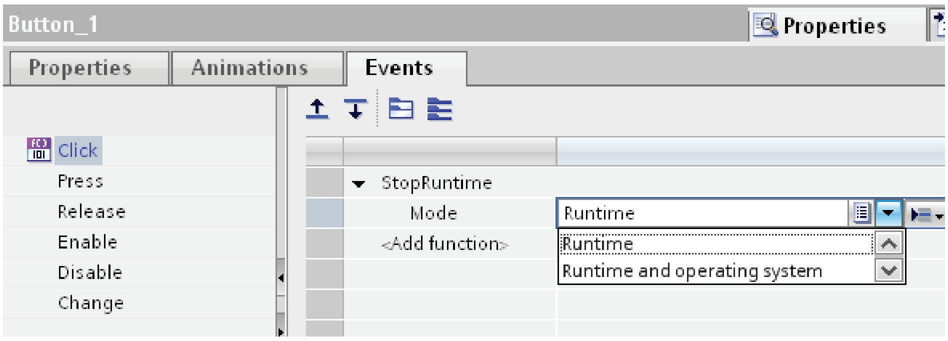
Proceed as follows to configure a function list:

1. Open the editor in WinCC in which the object is located.
2. Select the object.
3. Click "Properties > Events" in the Inspector window. Select the event for which you want to configure the function list.
4. Select the "<No Function>" entry in the drop-down list of the Inspector window.
5. Select the desired system function.
6. You can also enter the name of the system function.



The system function is entered in the function list.

7. If the system function has parameters, specify the values for the parameters.



8. If you want to add other system functions or functions to the function list, then repeat steps four to seven.

Result

The function list is configured. In addition, to the configured event, the status of the function list is displayed in the Inspector window. When the configured event occurs in Runtime, the function list is completed from top to bottom.

10.6.2.4 Editing a function list

Introduction

A function list can be edited as follows:

- Changing the order of execution for system functions
- Removing a system function

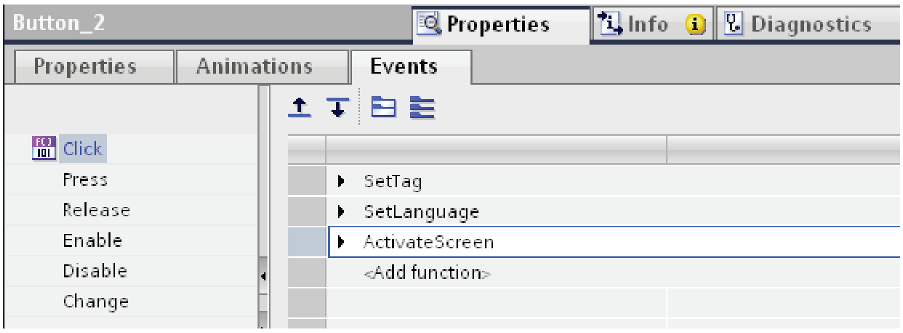
For additional information, refer to "Configuring a function list".

Requirement

The function list is configured.

Changing the order of a system function

1. Select the desired system function in the function list.
2. Then click the appropriate direction arrow in the inspector window until the system function or customized function is in the desired position.



Changing the order of several system functions

1. Hold down the <Shift> key.
2. Click the relevant system functions, working in succession.
3. Move the selection to the desired position by drag&drop.

Removing a system function

1. Select the desired system function in the function list.
2. Select "Delete" from the shortcut menu.

10.6.2.5 Executing a function list in Runtime

Principle

A function list is executed from top to bottom in Runtime. A distinction is made between synchronous and asynchronous execution, so that no waiting periods ensue during execution. The distinction is made by the system by evaluating the different runtimes of the system functions. User-defined functions are always executed synchronously independent of the runtime. If a system function returns an error status, the execution of the function list is cancelled.

Synchronous execution

During synchronous execution, the system functions in a function list are executed one after the other. The previous system function must be finished before the next system function can be executed.

Asynchronous execution

System functions that perform file operations such as saving and reading have a longer runtime than system functions that, for example, set a tag value.

Therefore, system functions with longer runtimes are executed asynchronously. For example, while a system function is writing a recipe data record to a storage medium, the next system function is already being executed. Due to the parallel execution of system functions, waiting periods at the HMI device are avoided.

10.6.3 Example

10.6.3.1 Changing the operating mode on the HMI device with the current display

Scheduled task

In this example, you use the "SetDeviceMode" system function to switch between the "online" and "offline" modes on the HMI device. You also display the current set operating mode on the HMI device.

Requirements

A screen has been created.

Settings

For this example you require a HMI-tag and a text list with the following settings:

HMI tag:

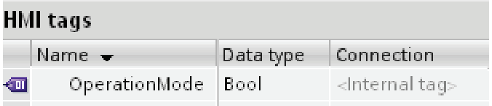
Name	PLC connection	Type
OperatingMode	No	Bool

Text list:

Name	Contains	Values
ShowOperatingMode	Bit (0/1)	1: Operating mode: "Online" 0: Operating mode: "Offline"

Procedure

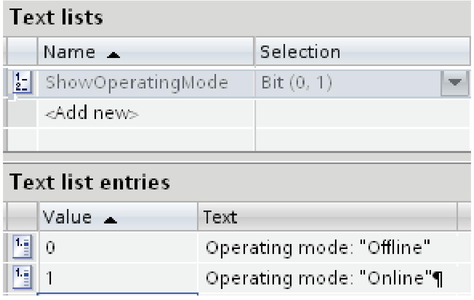
1. Create the "OperatingMode" HMI-tag indicated above.



HMI tags

Name	Data type	Connection
OperationMode	Bool	<Internal tag>

2. Create the "ShowOperatingMode" text list indicated above.



Text lists

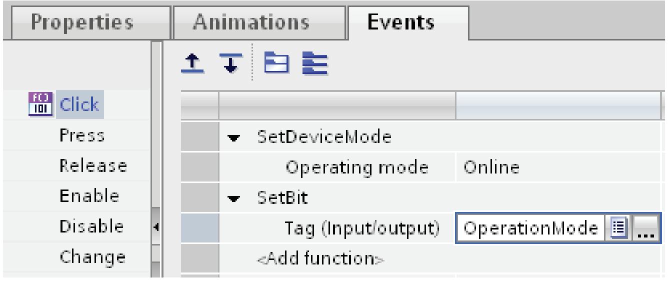
Name	Selection
ShowOperatingMode	Bit (0, 1)
<Add new>	

Text list entries

Value	Text
0	Operating mode: "Offline"
1	Operating mode: "Online"¶

3. Open the screen and insert a button for which you configure the operating mode change to "online".
4. Click "Properties> Events" in the Inspector window. Select the "Press" event.
5. Configure the "SetDeviceMode" system function for the "Press" event. The system function is found in the selection list under "Settings".
6. For the "Mode" parameter, select the "Online" entry.
7. Configure the system function "SetBit" on the event "Press". The system function is found in the selection list under "Bit processing".

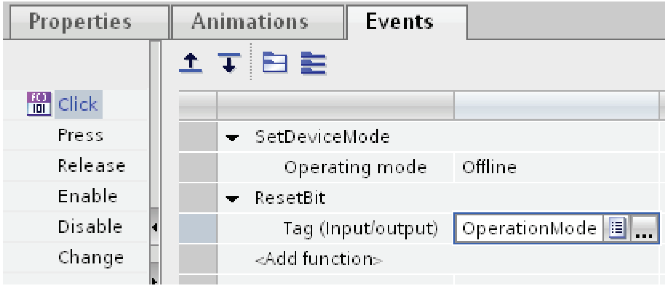
8. Select the HMI-tag "Operating mode" from the selection list for the parameter "Tag".



SetDeviceMode
Operating mode Online
SetBit
Tag (Input/output) OperationMode
<Add function>

9. Add a button in the process screen for which you configure the operating mode change to "offline".

10. Repeat steps four to seven. For the "Mode" parameter, select the "Offline" entry. Configure the system function "ResetBit" in place of the system function "SetBit."



SetDeviceMode
Operating mode Offline
ResetBit
Tag (Input/output) OperationMode
<Add function>

Interim result

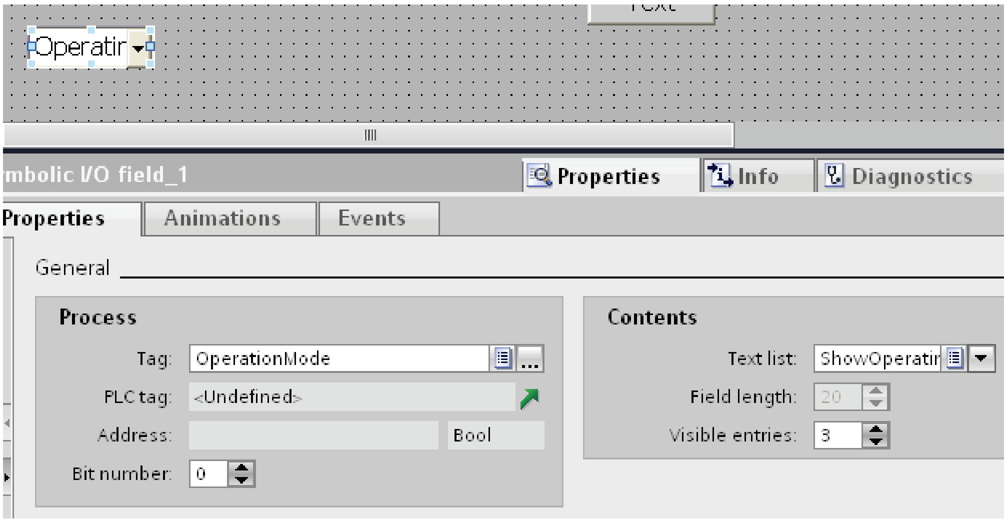
You can toggle the operating mode of the HMI device with the two buttons in Runtime.

You want to display the current set operating mode in an output field on the HMI device.

Procedure

1. Create a "Symbolic I/O field" in the process image. Click "Properties > Properties" in the Inspector window.

2. Make the following settings in the "General" group:
   - Select "Output" as the "Mode".
   - Select the text list "Show operating mode" as "Text list".
   - Select "Operating mode" as "Tag".



Tag: OperationMode
PLC tag: <Undefined>
Address: Bool
Bit number: 0
Text list: ShowOperatir
Field length: 20
Visible entries: 3

Result

When you change the operating mode with the buttons, the currently set operating mode on the HMI device is always shown.

10.6.4 Reference

10.6.4.1 Function list

Device-based dependency of system functions

Availability of system functions

The following table shows the availability of system functions and user-defined functions on the HMI devices.

Technical data subject to change.

Overview

	KP300 Basic PN	KTP400 Basic PN	KTP 600 Basic DP	KTP 600 Basic PN	KTP 1000 Basic DP	KTP 1000 Basic PN	TP 1500 Basic PN
User-defined functions	No	No	No	No	No	No	No
Logoff (Page 2297)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
AdjustContrast (Page 2298)	Yes	Yes	No	Yes [1)]	Yes	Yes	Yes
ActivateScreen (Page 2299)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ActivateScreenByNumber (Page 2300)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ActivateCleanScreen (Page 2301)	No	Yes	Yes	Yes	Yes	Yes	Yes
ActivatePreviousScreen (Page 2301)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
UpdateTag (Page 2302)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Logon (Page 2302)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
EditAlarm (Page 2303)	Yes	Yes	Yes	Yes	Yes	Yes	No
ScreenObjectCursorDown (Page 2304)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorUp (Page 2304)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectPageDown (Page 2306)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectPageUp (Page 2305)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
IncreaseFocusedValue (Page 2306)	Yes	Yes	Yes	Yes	Yes	Yes	No
IncreaseTag (Page 2307)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
GoToHome (Page 2308)	Yes	Yes	Yes	Yes	Yes	Yes	No
GoToEnd (Page 2308)	Yes	Yes	Yes	Yes	Yes	Yes	No
InvertBit (Page 2309)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
InvertBitInTag (Page 2310)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
InvertLinearScaling (Page 2311)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CalibrateTouchScreen (Page 2312)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewScrollForward (Page 2312)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewScrollBack (Page 2313)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewExtend (Page 2313)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewCompress (Page 2314)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewRulerRight (Page 2315)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewRulerLeft (Page 2315)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewSetRulerMode (Page 2316)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewStartStop (Page 2317)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewBackToBeginning (Page 2317)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
GetUserName (Page 2318)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
GetGroupNumber (Page 2319)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
GetPassword (Page 2319)	Yes	Yes	Yes	Yes	Yes	Yes	Yes

	KP300 Basic PN	KTP400 Basic PN	KTP 600 Basic DP	KTP 600 Basic PN	KTP 1000 Basic DP	KTP 1000 Basic PN	TP 1500 Basic PN
LinearScaling (Page 2320)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ClearAlarmBuffer (Page 2321)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ClearAlarmBufferProTool (Page 2322)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
AlarmViewUpdate (Page 2324)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
AlarmViewEditAlarm (Page 2323)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
AlarmViewAcknowledgeAlarm (Page 2324)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
AlarmViewShowOperatorNotes (Page 2325)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
AcknowledgeAlarm (Page 2325)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewNewDataRecord (Page 2326)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewGetDataRecordFromPLC (Page 2326)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewClearDataRecord (Page 2327)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewMenu (Page 2327)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewOpen (Page 2328)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewSetDataRecordToPLC (Page 2328)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewSaveDataRecord (Page 2329)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewSaveAsDataRecord (Page 2329)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewRenameDataRecord (Page 2330)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewShowOperatorNotes (Page 2330)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewBack (Page 2331)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ResetBit (Page 2331)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ResetBitInTag (Page 2332)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
PressButton (Page 2333)	Yes	Yes	Yes	Yes	Yes	Yes	No
ReleaseButton (Page 2334)	Yes	Yes	Yes	Yes	Yes	Yes	No
ShiftAndMask (Page 2334)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
PageDown (Page 2337)	Yes	Yes	Yes	Yes	Yes	Yes	No
PageUp (Page 2336)	Yes	Yes	Yes	Yes	Yes	Yes	No
SetDeviceMode (Page 2337)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SetBit (Page 2338)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SetBitInTag (Page 2339)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SetBitWhileKeyPressed (Page 2340)	Yes	Yes	Yes	Yes	Yes	Yes	Yes

	KP300 Basic PN	KTP400 Basic PN	KTP 600 Basic DP	KTP 600 Basic PN	KTP 1000 Basic DP	KTP 1000 Basic PN	TP 1500 Basic PN
SetColorBackgroundLighting (Page 2341)	Yes	No	No	No	No	No	No
SetLanguage (Page 2342)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SetTag (Page 2343)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SetConnectionMode (Page 2344)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SimulateSystemKey (Page 2345)	Yes	Yes	Yes	Yes	Yes	Yes	No
SimulateTag (Page 2346)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
StopRuntime (Page 2347)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TraceUserChange (Page 2348)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DecreaseFocusedValue (Page 2348)	Yes	Yes	Yes	Yes	Yes	Yes	No
DecreaseTag (Page 2349)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ChangeConnection (Page 2350)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ShowLogonDialog (Page 2352)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ShowOperatorNotes (Page 2352)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ShowAlarmWindow (Page 2353)	Yes	Yes	Yes	Yes	Yes	Yes	Yes

[1] For KTP600 Basic mono PN only.

System functions

Logoff

Description

Logs off the current user on the HMI device.

Use in the function list

Logoff

Use in user-defined functions

Logoff

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device-based dependency of system functions (Page 2294)".

Parameters

--

See also

Device-based dependency of system functions (Page 2294)

AdjustContrast

Description

Changes the contrast of the display one level on the HMI device.

Use in the function list

AdjustContrast (Adjust)

Use in user-defined functions

-

Parameters

Adjust

Specifies how the contrast is changed:

0 (hmiDecrease) = Decrease: Decreases the contrast one level.

1 (hmiIncrease) = Increase: Increases the contrast one level.

Application example

Objective

One button each for increasing and decreasing the screen contrast is desired.

Notes on configuring

Configure two buttons and configure the "AdjustContrast" system function on the "Press" event. The parameters "Increase" and "Decrease" are allocated.

Procedure on HMI device

When one of the two buttons is pressed in runtime, the contrast is increased or decreased one level.

See also

Device-based dependency of system functions (Page 2294)

ActivateScreen

Description

Performs a screen change to the given screen.

Use the "ActivateScreenByNumber" system function to change from the root screen to the permanent window or vice versa.

Use in the function list

ActivateScreen (Screen name, Object number)

Use in user-defined functions

ActivateScreen (Screen_name, Object_number)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Screen name

Name of the screen to which you change.

Object number

The operator control element which receives the focus in the given screen after the screen change. The number of the operator control element is to be determined using the tabulator sequence during configuration.

When "0" is specified:

- If the focus is in the permanent window when the system function is called up, the permanent window maintains the focus.
- If the focus is in the root screen when the system function is called up, the first operator control element in the given screen receives the focus.

Note

If the "Reach margin" event is assigned to the "ActivateScreen" system function, only the value "0" is valid for the "Object number" parameter. The active object is not defined by the object number, but rather by the X position it had prior to the screen change.

See also

Device-based dependency of system functions (Page 2294)

ActivateScreenByNumber (Page 2300)

ActivateScreenByNumber

Description

Performs a screen change to a screen depending on a tag value.

The screen is identified by its screen number.

Use in the function list

ActivateScreenByNumber (Screen number, Object number)

Use in user-defined functions

ActivateScreenByNumber (Screen_number, Object_number)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Screen number

Tag which contains the screen number of the destination screen.

When a change from the root screen to the permanent window is desired, "0" or "-1" is specified:

0 = Change from root screen to permanent window

-1 = Change from permanent window to root screen

Object number

The number of the screen object which receives the focus in the given screen after the screen change. The number of the operator control element is to be determined using the tabulator sequence during configuration.

When "0" is specified:

- If the focus is in the permanent window when the system function is called up, the permanent window maintains the focus.
- If the focus is in the root screen when the system function is called up, the first operator control element in the given screen receives the focus.

See also

Device-based dependency of system functions (Page 2294)

ActivateScreen (Page 2299)

ActivateCleanScreen

Description

Activates the clean screen on the HMI device. The display of the HMI device is disabled for the given time period.

When the display of the HMI device is deactivated, it can be cleaned without triggering touch functions by mistake.

Use in the function list

ActivateCleanScreen (Time period)

Use in user-defined functions

--

Parameters

Time period

Time period for which the display is disabled. The time remaining is displayed as a progress bar.

Value range in seconds from 10 through 300.

Note

The system function ActivateCleanScreen cannot be simulated.

ActivatePreviousScreen

Description

Performs a screen change to the screen which was activated before the current screen. The screen change is not performed if no screen was activated beforehand.

The last 10 screens that were called up are saved. A system alarm is output when you change to a screen which is no longer saved.

Note

If you want to use the system function, the screen to which you change has to be used in the navigation structure.

Use in the function list

ActivatePreviousScreen

Use in user-defined functions

ActivatePreviousScreen

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

--

See also

Device-based dependency of system functions (Page 2294)

UpdateTag

Description

Reads the current value of the tag with the specified Update ID from the PLC.

Use in the function list

UpdateTag (Update ID)

Use in user-defined functions

-

Parameters

Update ID

Update ID assigned to the tag that will be updated.

Logon

Description

Logs on the current user on the HMI device.

Use in the function list

Logon (Password, User name)

Use in user-defined functions

Logon (Password, User_name)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Password

The tag from which the password for the user logging on is read.

If the user is logged on, the password in the tag is deleted.

User name

The tag from which the user name for the user logging on is read.

See also

Device-based dependency of system functions (Page 2294)

EditAlarm

Description

Triggers the "Edit" event for all selected alarms.

If the alarms to be edited have not yet been acknowledged, the acknowledgment takes place automatically when this system function is called up.

This system function can only be used for function keys.

Use in the function list

EditAlarm

Use in user-defined functions

EditAlarm

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

--

See also

Device-based dependency of system functions (Page 2294)

ScreenObjectCursorUp

Description

Results in a line-by-line, upward cursor movement in the specified screen object.

The system function can be used for the following screen objects:

- User view
- Alarm view
- Recipe view

Use in the function list

ScreenObjectCursorUp (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the key function is triggered.

ScreenObjectCursorDown

Description

Results in a line-by-line, downward cursor movement in the specified screen object.

The system function can be used for the following screen objects:

- User view
- Alarm view
- Recipe view

Use in the function list

ScreenObjectCursorDown (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the key function is triggered.

ScreenObjectPageUp

Description

Results in a page-by-page, upward cursor movement in the specified screen object.

The system function can be used for the following screen objects:

- User view
- Alarm view
- Recipe view

Use in the function list

ScreenObjectPageUp (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the key function is triggered.

ScreenObjectPageDown

Description

Results in a page-by-page, downward cursor movement in the specified screen object.

The system function can be used for the following screen objects:

- User view
- Alarm view
- Recipe view

Use in the function list

ScreenObjectPageDown (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the key function is triggered.

IncreaseFocusedValue

Description

Adds the given value to the value of the tag which is connected to the input field (drop-down list, graphic selection list, slider bar) which has the current focus.

This system function can only be used for function keys.

Use in the function list

IncreaseFocusedValue (Value)

Use in user-defined functions

-

Parameters

Value

The value which is added to the tag value.

IncreaseTag

Description

Adds the given value to the value of the tags.

X = X + a

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, help tags must be used. The auxiliary tags can be assigned to the tag value with the "SetTag" system function.

If you configure the system function on the events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual value of the tags is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

IncreaseTag (Tag, Value)

Use in user-defined functions

IncreaseTag (Tag, Value)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Tag

The tag to which the given value is added.

Value

The value which is added.

See also

Device-based dependency of system functions (Page 2294)

SetTag (Page 2343)

GoToHome

Description

Executes the key function <Home> on the HMI device.

This system function is used when the HMI device does not have this functionality by default.

The system function can only be used for the following function keys.

Use in the function list

GoToHome

Use in user-defined functions

GoToHome

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

--

See also

Device-based dependency of system functions (Page 2294)

GoToEnd

Description

Executes the key function <End> on the HMI device.

This system function is used when the HMI device does not have this functionality by default.

The system function can only be used for the following function keys.

Use in the function list

GoToEnd

Use in user-defined functions

GoToEnd

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

--

See also

Device-based dependency of system functions (Page 2294)

InvertBit

Description

Inverts the value of the given tag of the "Bool" type:

- If the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

Use in the function list

InvertBit (Tag)

Use in user-defined functions

InvertBit (Tag)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Tag

The tag whose bit is set.

See also

Device-based dependency of system functions (Page 2294)

InvertBitInTag (Page 2310)

InvertBitInTag

Description

Inverts a bit in the given tag:

- If the bit in the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the bit in the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "InvertBit" system function instead.

Use in the function list

InvertBitInTag (Tag, Bit)

Use in user-defined functions

InvertBitInTag (Tag, Bit)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Tag

The tag in which the given bit is set.

Bit

The number of the bit that is set.

When this system function is used in a user-defined function, the bits in a tag are counted from right to left. The counting begins with 0.

See also

Device-based dependency of system functions (Page 2294)

InvertBit (Page 2309)

InvertLinearScaling

Description

Assigns a value to the tag X, which is calculated from the value of the given tag Y using the linear function X = (Y - b) / a.

The tags X and Y must not be identical. This system function is the inverse of the "LinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, a auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

InvertLinearScaling (X, Y, b, a)

Use in user-defined functions

InverseLinearScaling (X, Y, b, a)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

X

The tag which is assigned the value calculated from the linear equation.

Y

The tag whose value is used for the calculation.

b

The value which is subtracted.

a

The value through which is divided.

See also

Device-based dependency of system functions (Page 2294)

LinearScaling (Page 2320)

CalibrateTouchScreen

Description

Calls a program for calibrating the touch screen.

During the calibration process, there is a prompt to touch five positions on the screen display. Touch the screen display within 30 seconds, to confirm the calibration process. If the calibration is not completed within this time span, the calibration settings are discarded. The user prompt is in English.

Use this system function the first time you start the HMI device.

Use in the function list

CalibrateTouchScreen

Use in user-defined functions

CalibrateTouchScreen

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

--

Note

The CalibrateTouchScreen system function cannot be simulated.

See also

Device-based dependency of system functions (Page 2294)

TrendViewScrollForward

Description

Scrolls forward one display width in the Trend view.

Use in the function list

TrendViewScrollForward (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which is scrolled forward.

Note

The HMI device TP 177A does not support this system function for the "screen" object.

TrendViewScrollBack

Description

Scrolls back one display width to the left in the trend view.

Use in the function list

TrendViewScrollBack (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which is scrolled back.

Note

The HMI device TP 177A does not support this system function for the "screen" object.

TrendViewExtend

Description

Reduces the time period which is displayed in the trend view.

Use in the function list

TrendViewExtend (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the displayed time period is reduced.

Note

The HMI device TP 177A does not support this system function for the "screen" object.

TrendViewCompress

Description

Increases the time period which is displayed in the trend view.

Use in the function list

TrendViewCompress (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the displayed time period is increased.

Note

The HMI device TP 177A does not support this system function for the "screen" object.

TrendViewRulerRight

Description

Moves the read-line forwards (to the right) in the trend view.

Note

In order to be able to move the read-line, the read-line must have been switched on. This is done using the "TrendViewSetRulerMode" system function.

Use in the function list

TrendViewRulerLeft (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the read-line is moved forward.

Note

The HMI device TP 177A does not support this system function for the "screen" object.

See also

TrendViewSetRulerMode (Page 2316)

TrendViewRulerLeft

Description

Moves the read-line backwards (to the left) in the trend view.

Note

In order to be able to move the read-line, the read-line must have been switched on. This is done using the "TrendViewSetRulerMode" system function.

Use in the function list

TrendViewRulerRight (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the read-line is moved backwards.

Note

The HMI device TP 177A does not support this system function for the "screen" object.

See also

TrendViewSetRulerMode (Page 2316)

TrendViewSetRulerMode

Description

Hides or shows the read-line in the trend view. The read-line displays the Y value belonging to the X value.

Note

To ensure that the ruler is displayed, you have to activate the setting "Show ruler" in the trend view properties.

Use in the function list

TrendViewSetRulerMode (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the read-line is hidden or shown.

Note

The HMI device TP 177A does not support this system function for the "screen" object.

TrendViewStartStop

Description

Stops the trend recording or continues the trend recording in the trend view.

Use in the function list

TrendViewStartStop (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the recording of the trend is started or stopped.

Note

The HMI device TP 177A does not support this system function for the "screen" object.

TrendViewBackToBeginning

Description

Scrolls back to the beginning of the display range in the trend view.

Use in the function list

TrendViewBackToBeginning (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which you scroll to the beginning of the display range.

Note

The HMI device TP 177A does not support this system function for the "screen" object.

GetUserName

Description

Writes the user name of the user currently logged on to the HMI device in the given tag.

If the given tag has a control connection, the user name is also available in the PLC connection. This system function makes it possible, for example, to implement a user-dependent release of certain functionalities.

Use in the function list

GetUserName (Tag)

Use in user-defined functions

GetUserName (Tag)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Tag

The tag to which the user name is written.

See also

Device-based dependency of system functions (Page 2294)

GetGroupNumber

Description

Reads the number of the group to which the user logged on to the HMI device belongs, and writes it to the given tag.

Use in the function list

GetGroupNumber (Tag)

Use in user-defined functions

GetGroupNumber (Tag)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Tag

The tag to which the number of the group is written.

See also

Device-based dependency of system functions (Page 2294)

GetPassword

Description

Writes the password of the user currently logged on to the HMI device in the given tag.

Note

Make sure that the value of the given tag is not displayed in another place in the project.

Note

The passwords of SIMATIC Logon users cannot be read.

Use in the function list

GetPassword (Tag)

Use in user-defined functions

GetPassword (Tag)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Tag

The tag to which the password is written.

See also

Device-based dependency of system functions (Page 2294)

LinearScaling

Description

Assigns a value to the tag Y, which is calculated from the value of the given tag X using the linear function Y= (a *X) + b.

The inverse of this function is the "InvertLinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, a auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

LinearScaling (Y, a, X, b)

Use in user-defined functions

LinearScaling (Y, a, X, b)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Y

The tag which is assigned the value calculated from the linear equation.

a

The value with which is multiplied.

X

The tag whose value is used for the calculation.

b

The value which is added.

See also

Device-based dependency of system functions (Page 2294)

InvertLinearScaling (Page 2311)

ClearAlarmBuffer

Description

Deletes alarms from the alarm buffer on the HMI device.

Note

Alarms which have not yet been acknowledged are also deleted.

Use in the function list

ClearAlarmBuffer (Alarm class number)

Use in user-defined functions

ClearAlarmBuffer (Alarm_class_number)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Alarm class number

Determines which alarms are to be deleted from the alarm buffer:

0 (hmiAll) = All alarms/events

1 (hmiAlarms) = Alarms of alarm class "Errors"

2 (hmiEvents) = Alarms of alarm class "Warnings"

3 (hmiSystem) = Alarms of alarm class "System"

4 (hmiS7Diagnosis) = Alarms of alarm class "Diagnosis Events"

Note

Availability for specific devices

Alarms of alarm class "Diagnosis Events" are not available on Basic Panels.

See also

Device-based dependency of system functions (Page 2294)

ClearAlarmBufferProTool

Description

The system function exists to ensure compatibility.

It has the same functionality as the "ClearAlarmBuffer" system function, but uses the old ProTool numbering.

Use in the function list

ClearAlarmBufferProTool (Alarm class number)

Use in user-defined functions

ClearAlarmBufferProtoolLegacy (Alarm_class_number)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Alarm class number

Alarm class number whose messages are to be deleted:

-1 (hmiAllProtoolLegacy) = All alarms/events

0 (hmiAlarmsProtoolLegacy) = Alarms of alarm class "Errors"

1 (hmiEventsProtoolLegacy) = Alarms of alarm class "Warnings"

2 (hmiSystemProtoolLegacy) = Alarms of alarm class "System"

3 (hmiS7DiagnosisProtoolLegacy) = Alarms of alarm class "Diagnosis Events"

Note

Availability for specific devices

Alarms of alarm class "Diagnosis Events" are not available on Basic Panels.

See also

Device-based dependency of system functions (Page 2294)

ClearAlarmBuffer (Page 2321)

AlarmViewEditAlarm

Description

Triggers the event "Edit" for all alarms selected in the given alarm view.

This system function is used when the integrated button of the ActiveX control should not be used.

A system function can in turn be configured on the "Edit" event. For example, it is possible to change to the process screen in which the alarm appeared.

Note

If the alarms to be edited have not yet been acknowledged, the acknowledgment takes place automatically when this system function is called up.

Use in the function list

AlarmViewEditAlarm (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the alarm view in which the event is triggered.

Note

The HMI devices listed below do not support this system function for the "screen" object: OP 73, OP 77A, TP 177A.

AlarmViewUpdate

Description

Updates the enhanced alarm view.

Use in the function list

AlarmViewUpdate (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the alarm view which is updated.

AlarmViewAcknowledgeAlarm

Description

Acknowledges the alarms selected in the given alarm view.

This system function is used when the integrated button of the ActiveX control should not be used.

Use in the function list

AlarmViewAcknowledgeAlarm (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the alarm view in which the event is triggered.

Note

The HMI devices listed below do not support this system function for the "screen" object: OP 73, OP 77A, TP 177A.

AlarmViewShowOperatorNotes

Description

Displays the configured tooltip of the alarm selected in the given alarm view.

Use in the function list

AlarmViewShowOperatorNotes (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the alarm view in which the event is triggered.

Note

The HMI devices listed below do not support this system function for the "screen" object: OP 73, OP 77A, TP 177A.

AcknowledgeAlarm

Description

Acknowledges all selected alarms.

This system function is used when the HMI device does not have an ACK key or when the integrated key of the alarm view should not be used.

This system function can only be used for function keys.

Use in the function list

AcknowledgeAlarm

Use in user-defined functions

AcknowledgeAlarm

Can be used if the configured device supports user-defined functions. Additional information is available under "Auto-Hotspot".

Parameters

--

RecipeViewNewDataRecord

Description

Creates a new data record in the given recipe view.

Use in the function list

RecipeViewNewDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the new recipe data record is created.

RecipeViewGetDataRecordFromPLC

Description

Transfers the data record that is currently loaded in the PLC to the HMI device and displays it in the recipe view.

Use in the function list

RecipeViewGetDataRecordFromPLC (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the recipe data record from the PLC is displayed.

RecipeViewClearDataRecord

Description

Deletes the data record which is currently displayed in the recipe view.

Use in the function list

RecipeViewClearDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the displayed recipe data record is deleted.

RecipeViewMenu

Description

Opens the menu of the specified simple recipe view.

Only use this system function at a simple recipe view.

Use in the function list

RecipeViewMenu (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the menu is to be opened.

RecipeViewOpen

Description

Displays the data record values in the given recipe view or changes to the next selection field. The system function has no effect if the selection field for the recipe data record values is displayed on the HMI device.

Operation sequence of the selection lists in runtime:

- Recipe name
- Data record name
- RecipeDataRecordValues

This system function is used when a simple recipe view has been configured. In the simple recipe view, only one selection list is displayed at a time on the HMI device. Use the "RecipeViewBack" system function to display the previous selection list.

Use in the function list

RecipeViewOpen (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the command is triggered.

See also

RecipeViewBack (Page 2331)

RecipeViewSetDataRecordToPLC

Description

Transfers the recipe data record which is currently displayed in the recipe view to the PLC.

Use in the function list

RecipeViewSetDataRecordToPLC (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view from which the recipe data record is transferred to the connected PLC.

RecipeViewSaveDataRecord

Description

Saves the recipe data record which is currently displayed in the recipe view.

Use in the function list

RecipeViewSaveDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the recipe data record is saved.

RecipeViewSaveAsDataRecord

Description

Saves the data record currently being displayed in the recipe view under a new name.

Use in the function list

RecipeViewSaveAsDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Saves the data record currently being displayed in the recipe view under a new name and/or new number.

RecipeViewRenameDataRecord

Description

Renames the selected data record in the given recipe view.

Only use this system function at a simple recipe view.

Use in the function list

RecipeViewRenameDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the recipe data record is renamed.

RecipeViewShowOperatorNotes

Description

Displays the configured tooltip of the specified recipe view.

Use in the function list

RecipeViewShowOperatorNotes (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view whose configured tooltip is displayed.

RecipeViewBack

Description

Returns to the previous selection list in the given recipe view.

The system function has no effect if the selection field for the recipe is displayed on the HMI device. Operation sequence of the selection lists in runtime:

- Recipe name
- Data record name
- RecipeDataRecordValues

This system function is used when a simple recipe view has been configured. In the simple recipe view, only one selection list is displayed at a time on the HMI device. Use the "RecipeViewOpen" system function to display the recipe data record values or the next selection field.

Use in the function list

RecipeViewBack (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the command is triggered.

ResetBit

Description

Sets the value of a "Bool" type tag to 0 (FALSE).

Use in the function list

ResetBit (Tag)

Use in user-defined functions

ResetBit (Tag)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Tag

The BOOL type tag which is set to 0 (FALSE).

See also

Device-based dependency of system functions (Page 2294)

ResetBitInTag (Page 2332)

ResetBitInTag

Description

Sets a bit in the specified tag to 0 (FALSE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "ResetBit" system function instead.

Use in the function list

ResetBitInTag (Tag, Bit)

Use in user-defined functions

ResetBitInTag (Tag, Bit)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Tag

The tag in which a bit is set to 0 (FALSE).

Bit

The number of the bit that is set to 0 (FALSE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

See also

Device-based dependency of system functions (Page 2294)

ResetBit (Page 2331)

PressButton

Description

The system function can only be configured on the function keys of an HMI device and triggers the "Press key" event at the specified screen object.

Use this system function when you want to operate a button in a screen with a function key of the HMI device, for example.

Note

The "PressButton" and "ReleaseButton" system functions must always be configured together. If you configure the "PressButton" system function on the "Press key" event for a function key, the "ReleaseButton" system function is configured on the "Release" event for the same function key.

Use in the function list

PressButton (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object on which the event is triggered.

ReleaseButton

Description

The system function can only be configured on the function keys of an HMI device and triggers the "Release button" event at the specified screen object.

Use this system function when you want to operate a button in a screen with a function key of the HMI device, for example.

Note

The "PressButton" and "ReleaseButton" system functions must always be configured together. If you configure the "PressButton" system function on the "Press key" event for a function key, then the "ReleaseButton" system function is configured on the "Release key" event for the same function key.

Use in the function list

ReleaseButton (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object on which the event is triggered.

ShiftAndMask

Description

This system function converts the input bit pattern of the source tags into an output bit pattern of the target tags. This involves bit shifting and masking.

Note

If the source and target tag have a different number of bits, using the system function in the target tag can result in a violation of the value range.

Use in the function list

ShiftAndMask (Source tag, Target tag, Bits to shift, Bits to mask)

Use in user-defined functions

ShiftAndMask (Source_tag, Target_tag, Bits_to_shift, Bits_to_mask)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Source tag

The tag includes the input bit pattern. Integer-type tags, e.g. "Byte", "Char", "Int", "UInt", "Long" and "ULong" are permitted.

Example: The actual value 72 is set at the 16-bit integer source tag: 0000000001001000.

Target tag

The output bit pattern is saved in the tag. Integer type tags, e.g. "Byte", "Char", "Int", "UInt", "Long" and "ULong" are permitted.

Example: The shifted input bit pattern is multiplied by the bit mask, with bit-by-bit logical AND operation: 0000000000001001. The resultant decimal value "8" is saved to the target tag.

Please note the following:

- The source and target tags have the same number of bits.
- The number of bits to shift is less than the number of bits in the source tag and target tag.
- Bits to mask does not have more bits than the source tag and the target tag.

Bits to shift

Number of bits by which the input bit pattern is shifted right. A negative value shifts the input bit pattern to the left.

Example: "Bits to shift" has the value "+3". The input bit pattern is shifted right by three bits when the system function is called: 0000000000001001.
Bits to the left are padded with "0". Three bits are truncated on the right. The new decimal value is "9".

Note

The left bit is "1" in a source tag of the data type with negative signed integer. This sign bit is padded with "0" when the bits are shifted right. The sign changes to "+".

Bits to mask

An integer serves as bit mask. The bit pattern is used to multiply the shifted input bit pattern. Example: Integer "2478" with the bit pattern "0000100110101110".

You can enter the bit mask in three different ways:

- Hexadecimal: First enter the prefix "0h" or "0H", followed by an optional space for better readability. Then group the bit pattern in blocks of four (0000)(1001)(1010)(1110) and set each block in hexadecimal code: (0)(9)(A)(E). Only the characters 0-9, A-F, a-f are allowed: "0h 09AE".
- Binary: First enter the prefix "0b" or "0B", followed by an optional space for better readability. Then group the binary bit pattern into blocks of four 0000 1001 1010 1110 with spaces in between as a check. Only the characters "0" or "1" are allowed: "0b 0000 1001 1010 1110".
- Decimal: Enter the value "2478" directly, without a prefix.

See also

Device-based dependency of system functions (Page 2294)

PageUp

Description

Executes the key function <PageUp> on the HMI device.

This system function can only be used for function keys and tasks with a time trigger.

Use in the function list

PageUp

Use in user-defined functions

PageUp

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

-

See also

Device-based dependency of system functions (Page 2294)

PageDown

Description

Executes the key function <Pagedown> on the HMI device.

This system function can only be used for function keys.

Use in the function list

PageDown

Use in user-defined functions

PageDown

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

-

See also

Device-based dependency of system functions (Page 2294)

SetDeviceMode

Description

Toggles the operating mode on the HMI device. The following types of operation are possible: "Online", "Offline" and "Load".

Use in the function list

SetDeviceMode (Operating mode)

Use in user-defined functions

SetDeviceMode (Operating_mode)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Operating mode

Determines the operating mode of the HMI device:

0 (hmiOnline) = Online: The connection to the PLC is established.

1 (hmiOffline) = Offline: The connection to the PLC is disconnected.

2 (hmiTransfer) = load: A project can be transferred from the configuration computer to the HMI device.

Note

If you use a PC as an HMI device, the runtime software will be exited when you change operating mode after "Load".

See also

Device-based dependency of system functions (Page 2294)

SetConnectionMode (Page 2344)

SetBit

Description

Sets the value of a "Bool" type tag to 1 (TRUE).

Use in the function list

SetBit (Tag)

Use in user-defined functions

SetBit(Tag)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Tag

The BOOL type tag which is set to 1 (TRUE).

See also

Device-based dependency of system functions (Page 2294)

SetBitInTag

Description

Sets a bit in the given tag to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "SetBit" system function instead.

Use in the function list

SetBitInTag (Tag, Bit)

Use in user-defined functions

SetBitInTag(Tag, Bit)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Tag

The tag in which a bit is set to 1 (TRUE).

Bit

The number of the bit that is set to 1 (TRUE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

Note

The guaranteed update of the tags used with actual process values is absolutely conditional in terms of reliable functionality. You should therefore configure the tag in an I/O field or assign the system function to a screen object, such as a button.

If you have configured a short event such as the activation of an alarm for the system function you can only access the actual process values by setting the tag for continuous reading.

See also

Device-based dependency of system functions (Page 2294)

SetBit (Page 2338)

SetBitWhileKeyPressed

Description

Sets the bit of the given tag to 1 (TRUE) as long as the user keeps the configured key pressed.

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC. You should only access tags of the BOOL type with this system function to avoid problems with overlapping access to the same tag.

Note

All functions on the event "Release" are performed immediately by means of a screen change configured for a key, even if the key is kept pressed.

If the "SetBitWhileKeyPressed" system function is configured for a function key, the bit will be reset immediately following a screen change. This action is necessary since the key assignments change after the screen change.

If the PLC supports BOOL tags, do not use this system function. Use the "SetBit" system function instead.

Use in the function list

SetBitWhileKeyPressed (Tag, Bit)

Use in user-defined functions

-

Parameters

Tag

The tag in which a bit is temporarily set to 1 (TRUE). Use only tags of the type BOOL, as far as allowed by the PLC.

Bit

The number of the bit that is temporarily set to 1 (TRUE).

Note

The guaranteed update of the tags used with actual process values is absolutely conditional in terms of reliable functionality. You should therefore configure the tag in an IO field, or assign the function to a screen element such as a button.

If you configured a short event such as the activation of an alarm for the function you can only access the actual process values by setting the tag for continuous reading.

See also

SetBit (Page 2338)

SetColorBackgroundLighting

Description

Defines the background lighting of the button.

Note

The configuration that was set at Switch off is reestablished when restarting the HMI device.

Use in the function list

SetColorBackgroundLighting (Value)

Use in user-defined functions

-

Parameters

Value

Defines the background lighting of the button:

0 (hmiWhite) = White: No color

1 (hmiGreen) = Green: Green color

2 (hmiYellow) = Yellow: Yellow color

3 (hmiRed) = Red: Red color

SetLanguage

Description

Toggles the language on the HMI device. All configured text and system events are displayed on the HMI device in the newly set language.

Use in the function list

SetLanguage (Language)

Use in user-defined functions

SetLanguage(Language)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Language

Determines which language is set on the HMI device. The following specifications are possible:

- -1 (hmiToggle) = Toggle: Changes to the next language. The sequence is determined during configuration in the "Project languages" editor.
- Number you have defined under "Languages and fonts" in the "Runtime Settings" editor. Changes to the language with the given number.
- Language you have defined under "Languages and fonts" in the "Runtime Settings" editor.
- Language abbreviation in accordance with the VBScript5 reference: This changes to the language corresponding to the specified language code, e.g. "de-DE" for German (Germany) or "en-US" for English (United States).

  An overview of the language abbreviations is available in the basic information of VBScript under the topic "Area diagram-ID (LCID) Diagram".

See also

Device-based dependency of system functions (Page 2294)

SetTag

Description

Assigns a new value to the given tag.

Note

This system function can be used to assign strings and numbers, depending on the type of tag.

Use in the function list

SetTag (Tag, Value)

Use in user-defined functions

SetTag(Tag, Value)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Tag

The tag to which the given value is assigned.

Value

The value which the given tag is assigned.

Note

The "SetTag" system function is only executed after a connection has been established.

See also

Device-based dependency of system functions (Page 2294)

IncreaseTag (Page 2307)

SetConnectionMode

Description

Connects or disconnects the given connection.

Note

A connection to the PLC cannot be established until the operating mode ONLINE has been set on the HMI device. Use the "SetDeviceMode" system function for this purpose.

Use in the function list

SetConnectionMode (Mode, Connection)

Use in user-defined functions

SetConnectionMode(Mode, Connection)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Mode

Determines whether a connection to the PLC is established or disconnected:

0 (hmiOnline) = Online: Connection is established.

1 (hmiOffline) = Offline: Connection is disconnected.

Connection

The PLC to which the HMI device is connected. You specify the name of the PLC in the connection editor.

Multiple use of the system function in a user-defined function

If you use the "SetConnectionMode" system function for different connections, it may be possible that not all system functions are executed correctly. Proceed as follows to prevent this situation:

1. Create a "BOOL" type tag with the start value "0".
2. Configure the "SetConnectionMode" system function on the "Value change" event of the HMI tags. If you want to disconnect three connections, for example, you must configure the system function three times.
3. In the user-defined function, apply the "InvertBit" system function to the HMI tag.

Application example

Two typical application examples for this system function are as follows:

- Test

  As long as no PLC is connected to the HMI device, no error messages will be output during the test on the HMI device. If the HMI device is connected to a PLC, the connection to the PLC can be established by pressing a key.

- Start up

  Several PLCs are to be configured for a system. At first, all PLCs except one are configured "Offline". After start up of the first PLC, the connection to each of the other PLCs is established by pressing a key. In this way, the other PLCs are started up one after another.

See also

Device-based dependency of system functions (Page 2294)

SetDeviceMode (Page 2337)

SimulateSystemKey

Description

Simulates the behavior of a System Key. Use this system function if a system key, such as the "ACK" key, "Input" key or the number pad is not available on the HMI device.

Use in the function list

SimulateSystemKey (System key)

Use in user-defined functions

-

Parameters

System Key

System Key, the behavior for which is to be simulated.

System key "+/-"

With the SimulateSystemKey system function, the system key "+/-" is only supported for the following HMI devices:

- KTP400 Basic mono PN
- KTP600 Basic mono PN
- KTP600 Basic color PN
- KTP600 Basic color DP
- KTP1000 Basic PN
- KTP1000 Basic DP
- KTP1500 Basic PN

Use the system keys "+" and "-" separately for all other HMI devices.

SimulateTag

Description

Simulates the behavior of tags and dynamic objects such as text lists, without having the HMI device connected to a PLC.

This system function is used, for example, to demonstrate the functionality of a project.

NOTICE
Only tags of the data type Integer can be used for simulation. Tags of the data types Integer and Double Integer, however, can be used with OP 73, OP 77A, TP 177A.

Use in the function list

SimulateTag (Tag, Cycle, Maximum value, Minimum value, Value)

Use in user-defined functions

-

Parameters

Tag

The tag whose value is changed.

Cycle

The factor by which the basic cycle of 200 milliseconds is multiplied. The cycle defines when the tag value is changed by the specified value. Possible cycles between 1 and 32767.

Maximum value

The maximum value that the tag can assume during simulation. The maximum value must be greater than the minimum value but less than / equal to 32767.

Minimum value

The minimum value that the tag can assume during simulation. The minimum value must be greater than the maximum value but less than / equal to -32768.

Value

The value by which the tag value is changed during each cycle. Possible values between -32768 and 32767.

- A positive value increases the tag value. When the maximum value is reached, the tag value is set to the minimum value after the next update cycle.
- A negative value reduces the tag value. When the minimum value is reached, the tag value is set to the maximum value after the next update cycle.

StopRuntime

Description

Exits the runtime software and thereby the project running on the HMI device.

Use in the function list

StopRuntime (Mode)

Use in user-defined functions

StopRuntime(Mode)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Mode

Determines whether the operating system is shut down after exiting runtime.

0 (hmiStopRuntime) = Runtime: Operating system is not shut down

1 (hmiStopRuntimeAndOperatingSystem) = Runtime and operating system: The operating system is shut down (not possible with WinCE)

See also

Device-based dependency of system functions (Page 2294)

TraceUserChange

Description

Outputs a system event that shows which user is currently logged in on the HMI device.

This system function can only be used in the Scheduler.

Use in the function list

TraceUserChange

Use in user-defined functions

-

Parameters

--

DecreaseFocusedValue

Description

Subtracts the specified value from the value of the tag which is connected to the screen object and currently has the focus.

The system function can be configured:

- Input field
- Symbolic selection list
- Graphic selection list
- Slider bar

Use in the function list

DecreaseFocusedValue (Value)

Use in user-defined functions

-

Parameters

Value

The value which is subtracted from the tag value.

DecreaseTag

Description

Subtracts the given value from the tag values.

X = X - a

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, help tags must be used. The auxiliary tags are assigned to the tag value with the "SetTag" system function.

If you configure the system function on the events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual value of the tags is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

DecreaseTag (Tag, Value)

Use in user-defined functions

DecreaseTag(Tag, Value)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Tag

The tag from which the given value is subtracted.

Value

The value which is subtracted.

See also

Device-based dependency of system functions (Page 2294)

ChangeConnection

Description

Disconnects the connection to the currently used PLC in use and establishes a connection to a PLC with another address. The newly connected PLC must belong to the same device class (S7-1200, S7-300, ...etc). With S7-1200, the use of the function is also only permitted for absolute addressing.

Note

When changing to another address, ensure that the address is not already being used by another HMI device.

The follows address types are supported:

- IP address
- MPI address

The follows PLC types are supported:

- SIMATIC S7 1200
- SIMATIC S7 300/400
- SIMATIC S7-NC
- SIMOTION

Use in the function list

ChangeConnection (Connection, Address, Slot, Rack)

Use in user-defined functions

ChangeConnection (Connection, Address, Slot, Rack)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Connection

Name of the connection that is disconnected. The name is set during configuration, for example, in the "Connections" editor.

Address

MPI/PROFIBUS or IP address of the PLC to which the connection will be established.

Note

Set the address by means of a tag. The objects list displays the tags of all data types. Select only tags of the following data types:

- Ethernet connection: "String" data type
- MPI connection: "Int" data type

Slot

Slot of the PLC to which the connection will be established.

Rack

Rack of the PLC to which the connection will be established.

Application example

You want to operate one HMI device on several machines. Configure the necessary PLCs in the project, to which you want to change by pressing a key. When changing the PLC, the connection to the PLC in use is disconnected. Then the connection to the new PLC with other address parameters is reestablished. To access the values of the new PLC, configure the same tags for the PLC used.

The PLC which you have indicated when creating the project will be used as default.

1. Enter the name and address of the PLC in the "Connections" editor.
2. Configure a button in the process screen.
3. Configure the "ChangeConnection" system function on the "Press" event.
4. Enter the name of the connection and address of the PLC as parameters.

See also

Device-based dependency of system functions (Page 2294)

ShowLogonDialog

Description

Opens a dialog on the HMI device with which the user can log on to the HMI device.

Use in the function list

ShowLogonDialog

Use in user-defined functions

-

Parameters

--

ShowOperatorNotes

Application

Displays the configured tooltip of the selected object.

If the system function is configured on a function key, the tooltip for the screen object that currently has the focus is displayed. If a tooltip is configured for the screen itself, you can switch to this text by pressing <Enter> or by double-clicking on the help window.

If the system function is configured on a button, only the tooltip for the current screen is displayed. If a tooltip is configured on the button itself, initially only the tooltip for the button is displayed. You can press <Enter> or double-click on the help window to switch to the tooltip for the current screen.

Note

No other screen object can be used while the help window is open. To use the screen object, close the help window.

Closing the help window

You can close the help window in the following ways:

Using the keys:

- By pressing the <HELP> key again
- By pressing the <ESC> key

Using the touch screen:

- By pressing the ☒ button

Use in the function list

ShowOperatorNotes (Display mode)

Use in user-defined functions

ShowOperatorNotes (Display_mode)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Display mode

Determines whether the configured tooltip is hidden or shown:

0 (hmiOff) = Off: Configured tooltip is hidden

1 (hmiOn) = On: Configured tooltip is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes

See also

Device-based dependency of system functions (Page 2294)

ShowAlarmWindow

Description

Hides or shows the alarm window on the HMI device.

Use in the function list

ShowAlarmWindow (Object name, Display mode)

Use in user-defined functions

ShowAlarmWindow(Object_name, Display_mode)

Can be used if the configured device supports user-defined functions. Additional information is available under "Device-based dependency of system functions (Page 2294)".

Parameters

Object name

Name of the alarm view which is hidden or shown.

Display mode

Determines whether the alarm window is hidden or shown:

0 (hmiOff) = Off: Alarm view is hidden

1 (hmiOn) = On: Alarm view is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes

See also

Device-based dependency of system functions (Page 2294)

10.6.4.2 Events

Overview

Editors

Introduction

The following table shows which events occur in which editor.

Technical data subject to change.

Icon	Editor
📁	Screens
✉	HMI alarms
▤	HMI tags
5	Scheduler

	📁	✉	📋	5
Cleared (Page 2360)	√	--	--	--
Enable (Page 2361)	--	--	--	--
Adjust (Page 2361)	--	--	--	--
Loaded (Page 2361)	√	--	--	--
Execute (Page 2362)	--	--	--	√
Selection changed (Page 2362)	--	--	--	--
When dialog is opened (Page 2362)	--	--	--	√
When dialog is closed (Page 2362)	--	--	--	√
User change (Page 2363)	--	--	--	√
Screen change (Page 2363)	--	--	--	√
Disabling (Page 2364)	--	--	--	√
Double-click (Page 2364)	--	--	--	--
Pressing (Page 2365)	--	--	--	--
On Finish Input (Page 2365)	--	--	--	--
Press ESC twice (Page 2365)	--	--	--	--
Outgoing (Page 2366)	--	√	--	--
Incoming (Page 2366)	--	√	--	--
Click (Page 2366)	--	--	--	--
Loop-in alarm (Page 2367)	--	√	--	--
Releasing (Page 2367)	--	--	--	--
In the event of high limit violation (Page 2367)	--	--	√	--
Alarm buffer overflow (Page 2368)	--	--	--	√
In the event of low limit violation (Page 2368)	--	--	√	--
Acknowledging (Page 2368)	--	√	--	--
Reach margin (Page 2369)	--	--	--	--
Runtime stop (Page 2369)	--	--	--	√
Press key (Page 2370)	--	--	--	--
Release key (Page 2370)	--	--	--	--
Overflow (Page 2370)	--	--	--	--
Switch OFF (Page 2371)	--	--	--	--
Switch ON (Page 2371)	--	--	--	--
Low free storage space (Page 2371)	--	--	--	--
Free space critically low (Page 2371)	--	--	--	--
Value change (Page 2372)	--	--	√	--
Time expired (Page 2372)	--	--	--	--

Basic objects

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

Icon	Object
╱	Line
⬬	Ellipse
●	Circle
▬	Rectangle
A	Text field
🖼	Graphic view

	╱	⬬	●	▬	A	🖼
Cleared (Page 2360)	--	--	--	--	--	--
Enable (Page 2361)	--	--	--	--	--	--
Adjust (Page 2361)	--	--	--	--	--	--
Loaded (Page 2361)	--	--	--	--	--	--
Execute (Page 2362)	--	--	--	--	--	--
Selection changed (Page 2362)	--	--	--	--	--	--
When dialog is opened (Page 2362)	--	--	--	--	--	--
When dialog is closed (Page 2362)	--	--	--	--	--	--
User change (Page 2363)	--	--	--	--	--	--
Screen change (Page 2363)	--	--	--	--	--	--
Disabling (Page 2364)	--	--	--	--	--	--
Double-click (Page 2364)	--	--	--	--	--	--
Pressing (Page 2365)	--	--	--	--	--	--
On Finish Input (Page 2365)	--	--	--	--	--	--
Press ESC twice (Page 2365)	--	--	--	--	--	--
Outgoing (Page 2366)	--	--	--	--	--	--
Incoming (Page 2366)	--	--	--	--	--	--
Click (Page 2366)	--	--	--	--	--	--
Loop-in alarm (Page 2367)	--	--	--	--	--	--
Releasing (Page 2367)	--	--	--	--	--	--
In the event of high limit violation (Page 2367)	--	--	--	--	--	--
Alarm buffer overflow (Page 2368)	--	--	--	--	--	--

	/	⬬	●	■	A	🖼
In the event of low limit violation (Page 2368)	--	--	--	--	--	--
Acknowledging (Page 2368)	--	--	--	--	--	--
Reach margin (Page 2369)	--	--	--	--	--	--
Runtime stop (Page 2369)	--	--	--	--	--	--
Press key (Page 2370)	--	--	--	--	--	--
Release key (Page 2370)	--	--	--	--	--	--
Overflow (Page 2370)	--	--	--	--	--	--
Switch OFF (Page 2371)	--	--	--	--	--	--
Switch ON (Page 2371)	--	--	--	--	--	--
Low free storage space (Page 2371)	--	--	--	--	--	--
Free space critically low (Page 2371)	--	--	--	--	--	--
Value change (Page 2372)	--	--	--	--	--	--
Time expired (Page 2372)	--	--	--	--	--	--

Elements

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

Icon	Object
0.12	IO field
▭	Button
▾	Symbolic IO field
🖼	Graphic IO field
5	Date/time field
▮	Bar
0 I	Switch

	0.12	▭	▾	🖼	5	▮	0 I
Cleared (Page 2360)	--	--	--	--	--	--	--
Enable (Page 2361)	√	√	√	√	--	--	√
Adjust (Page 2361)	--	√	√	--	--	--	√
Loaded (Page 2361)	--	--	--	--	--	--	--
Execute (Page 2362)	--	--	--	--	--	--	--
Selection changed (Page 2362)	--	--	--	--	--	--	--

	0.12				5		0 I
When dialog is opened (Page 2362)	--	--	--	--	--	--	--
When dialog is closed (Page 2362)	--	--	--	--	--	--	--
User change (Page 2363)	--	--	--	--	--	--	--
Screen change (Page 2363)	--	--	--	--	--	--	--
Disabling (Page 2364)	√	√	√	√	--	--	√
Double-click (Page 2364)	--	--	--	--	--	--	--
Pressing (Page 2365)	--	√	--	--	--	--	--
On Finish Input (Page 2365)	--	--	--	--	--	--	--
Press ESC twice (Page 2365)	--	--	--	--	--	--	--
Outgoing (Page 2366)	--	--	--	--	--	--	--
Incoming (Page 2366)	--	--	--	--	--	--	--
Click (Page 2366)	--	√	--	--	--	--	--
Loop-in alarm (Page 2367)	--	--	--	--	--	--	--
Releasing (Page 2367)	--	√	--	--	--	--	--
Auto-Hotspot	--	--	--	--	--	--	--
Alarm buffer overflow (Page 2368)	--	--	--	--	--	--	--
In the event of low limit violation (Page 2368)	--	--	--	--	--	--	--
Acknowledging (Page 2368)	--	--	--	--	--	--	--
Reach margin (Page 2369)	--	--	--	--	--	--	--
Runtime stop (Page 2369)	--	--	--	--	--	--	--
Press key (Page 2370)	--	--	--	--	--	--	--
Release key (Page 2370)	--	--	--	--	--	--	--
Overflow (Page 2370)	--	--	--	--	--	--	--
Switch OFF (Page 2371)	--	--	--	--	--	--	√
Switch ON (Page 2371)	--	--	--	--	--	--	√
Low free storage space (Page 2371)	--	--	--	--	--	--	--
Free space critically low (Page 2371)	--	--	--	--	--	--	--
Value change (Page 2372)	--	--	--	--	--	--	--
Time expired (Page 2372)	--	--	--	--	--	--	--

See also

In the event of high limit violation (Page 2367)

Controls

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

Icon	Object
⚠	Alarm view/alarm window
⚠	Alarm indicator
📈	Trend view
👤	User view
🗃	Recipe view
ℹ	Auxiliary indicator

	⚠	⚠	📈	👤	🗃	ℹ
Cleared (Page 2360)	--	--	--	--	--	--
Enable (Page 2361)	√	--	√	√	--	--
Adjust (Page 2361)	--	--	--	--	--	--
Loaded (Page 2361)	--	--	--	--	--	--
Execute (Page 2362)	--	--	--	--	--	--
Selection changed (Page 2362)	--	--	--	--	--	--
When dialog is opened (Page 2362)	--	--	--	--	--	--
When dialog is closed (Page 2362)	--	--	--	--	--	--
User change (Page 2363)	--	--	--	--	--	--
Screen change (Page 2363)	--	--	--	--	--	--
Disabling (Page 2364)	√	--	√	√	--	--
Double-click (Page 2364)	--	--	--	--	--	--
Pressing (Page 2365)	--	--	--	--	--	--
On Finish Input (Page 2365)	--	--	--	--	--	--
Press ESC twice (Page 2365)	--	--	--	--	--	--
Outgoing (Page 2366)	--	--	--	--	--	--
Incoming (Page 2366)	--	--	--	--	--	--
Click (Page 2366)	--	√	--	--	--	--
Loop-in alarm (Page 2367)	--	√	--	--	--	--
Releasing (Page 2367)	--	--	--	--	--	--
In the event of high limit violation (Page 2367)	--	--	--	--	--	--
Alarm buffer overflow (Page 2368)	--	--	--	--	--	--
In the event of low limit violation (Page 2368)	--	--	--	--	--	--

		⚠				ℹ
Acknowledging (Page 2368)	--	--	--	--	--	--
Reach margin (Page 2369)	--	--	--	--	--	--
Runtime stop (Page 2369)	--	--	--	--	--	--
Press key (Page 2370)	--	--	--	--	--	--
Release key (Page 2370)	--	--	--	--	--	--
Overflow (Page 2370)	--	--	--	--	--	--
Switch OFF (Page 2371)	--	--	--	--	--	--
Switch ON (Page 2371)	--	--	--	--	--	--
Low free storage space (Page 2371)	--	--	--	--	--	--
Free space critically low (Page 2371)	--	--	--	--	--	--
Value change (Page 2372)	--	--	--	--	--	--
Time expired (Page 2372)	--	--	--	--	--	--

Events

Cleared

Description

Occurs when the active screen on the HMI device is cleared.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Enable

Description

Occurs when the user selects a display or operating object using the configured tab sequence.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Note

If the user e.g. clicks a button with the mouse, the "Click" event is triggered. Users wishing to trigger the "Enable" event must select the button using the tab key.

The "Activate" event is only used to detect whether an object was selected. The event does not trigger a password prompt.

For this reason, do not use the "Activate" event if you want to configure access protection on the function call of the object.

Adjust

Description

Occurs if the status of a display and operator control object changes.

The status of an object changes if, for example, the user presses the key.

Note

Please note that the availability of the event depends on the HMI device and object type.

Loaded

Description

Occurs when all configured display and operating objects are loaded in the active screen after a screen change.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Note

Enable a screen change to ensure that the connection with the control is established after switch-on.

Execute

Description

Occurs when the scheduled task has been executed.

Selection changed

Description

Occurs when the user changes the selection.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

When dialog is opened

Description

The event is triggered when a modal dialog opens.

Note

Please note that the availability of the event depends on the HMI device and object type.

When dialog is closed

Description

The event is triggered when a modal dialog closes.

Note

Please note that the availability of the event depends on the HMI device and object type.

User change

Description

Occurs when a user logs off at an HMI device or another user logs on at the HMI device.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Screen change

Description

Occurs when all configured display and operating objects are loaded in the screen after a screen change.

Use the "Loaded" event if you want to perform other system functions during a screen change to a certain screen.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Disabling

Description

Occurs when the user takes the focus from a display and operating object.

A screen object can be disabled using the configured tab order or by performing another action with the mouse.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Note

System functions or user-defined functions on the "Deactivate" event of a screen are not executed when a screen is being closed.

The "Deactivate" event is only used to detect whether an object was deselected. The event does not trigger a password prompt.

For this reason, do not use the "Deactivate" event if you want to configure access protection on the function call of the object.

Double-click

Description

Occurs when the user double-clicks on an object from the symbol library.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Pressing

Description

Occurs when the user clicks on a button with the left mouse button, presses <RETURN> or <SPACE>.

Also occurs when the user right-clicks on an object of the symbol library.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

On Finish Input

Description

Triggered when you confirm input at an I/O field by pressing ENTER, or by mouse click, or by touch screen operation.

The "On Finish Input" event is also started if the value of a tag does not change, for example, if a value is exceeded, or if a user cancels the dialog to acknowledge a tag (Audit option package) that has to be acknowledged.

The event is not triggered, on the other hand, by user logon or by input fields configured with an authorization.

Note

Please note that the availability of the event depends on the HMI device and object type.

Press ESC twice

Description

Occurs when the user presses the <ESC> key twice at the HMI device.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Outgoing

Description

Occurs when an alarm is deactivated.

Note

Please note that the availability of the event depends on the HMI device and object type.

Incoming

Description

Occurs when an alarm is triggered and displayed in the alarm view.

Note

Please note that the availability of the event depends on the HMI device and object type.

Click

Description

Occurs if the user clicks a display and operating object with the mouse or touches the touch display with a finger.

In case you click the incorrect object, prevent processing of configured function list as follows:

- Move the mouse pointer away from the object while keeping the mouse button pressed. Release the mouse button as soon as the mouse pointer leaves the object. The function list will then not be processed.
- On touch displays, the display must be touched with the finger until a reaction occurs, e.g., a screen change.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Loop-in alarm

Description

Occurs as soon as the user selects an alarm in the alarm view and then clicks on the "Loop-In-Alarm" button or double clicks on the alarm.

For the "Loop-In-Alarm" event, you configure system functions, such as a change to the screen in which the alarm occurred.

You cannot configure local scripts for the "Loop-In-Alarm" event in Runtime Professional.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Releasing

Description

Occurs when the user releases a button.

This even does not occur, as long as the button remains pressed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

In the event of high limit violation

Description

Occurs when the high limit of a tag is exceeded.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Alarm buffer overflow

Description

Occurs when the configured size of the alarm buffer is reached.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

In the event of low limit violation

Description

Occurs when the low limit of a tag is undershot.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Acknowledging

Description

Occurs when the user acknowledges an alarm.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Reach margin

Description

Occurs when the user reaches the beginning or the end of a scrollable area.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Note

A user-defined function must not be configured for the "Boundary reached" event.

Configurable objects

The event can only be configured on the <Up> and <Down> keys, or on the keys on which you have configured the "ScreenObjectPageUp" or "ScreenObjectPageDown" system functions.

Runtime stop

Description

Occurs when the user exits the Runtime software on the HMI device.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Note

A user-defined function must not be configured for the "Runtime stop" event.

Press key

Description

Occurs when the user presses a function key.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Release key

Description

Occurs when the user releases a function key.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Overflow

Description

Occurs when the configured size of the log is reached. You use the log type "Trigger event".

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Switch OFF

Description

Occurs when the user moves the display and operating object "Switch" to the OFF position.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Switch ON

Description

Occurs when the user moves the display and operating object "Switch" to the ON position.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Low free storage space

Description

This event is triggered if the storage space available on the medium to which the Audit Trail is less than the configured minimum.

Free space critically low

Description

This event is triggered if the storage medium to which an Audit Trail is saved provides insufficient storage space due to hardware restrictions.

Value change

Description

Occurs when the value of an object or the value of an array element changes.

The value change of a tag is triggered by the PLC or by the user, e.g. when a new value is entered.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Time expired

Description

Occurs when the time configured in the scheduler expires.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

10.7 Planning tasks

10.7.1 Field of application of the Scheduler

Definition

You can use the Scheduler to configure tasks to run independent of the screen in the background. You create tasks by linking system functions or scripts to a trigger. The linked functions will be called when the triggering event occurs.

Example of an application

The Scheduler is used to execute event-controlled tasks automatically. For example, you use a task to automate the following:

- Regular swap out of log data
- Printout of an alarm report when an alarm buffer overflow occurs
- Printout of a report at shift end
- Monitoring a tag
- Monitoring of a user change

Note

The availability of the listed examples is determined by the HMI device.

See also

Working with tasks and triggers (Page 2374)

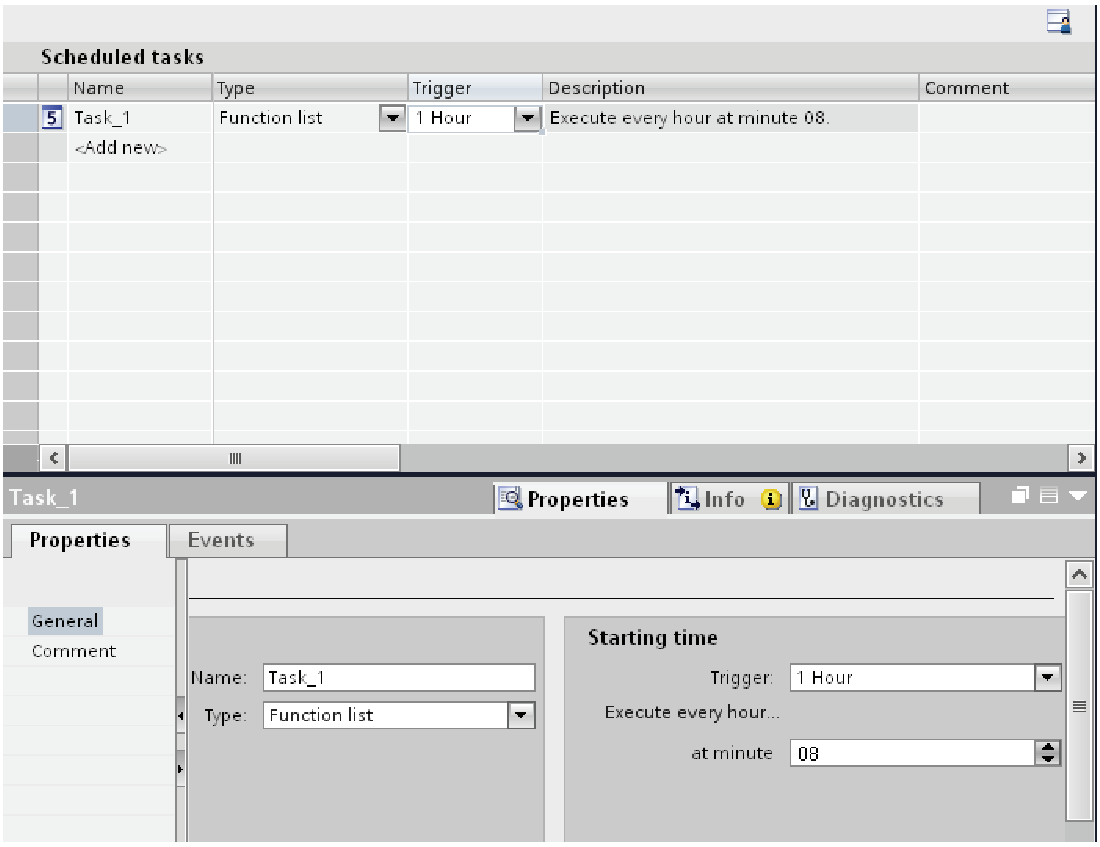
Example: Update user following change of user (Page 2379)

Work area of the "Scheduler" editor (Page 2375)

10.7.2 Working with tasks and triggers

Introduction

A task consists of a trigger and a task type.



Scheduled tasks

Name	Type	Trigger	Description	Comment
Task_1	Function list	1 Hour	Execute every hour at minute 08.	
<Add new>				

Task_1

Name: Task_1
Type: Function list

Starting time

Trigger: 1 Hour
Execute every hour...
at minute 08

Starting a task

Controlled by a trigger, the Scheduler starts the task linked to the trigger.

See also

Field of application of the Scheduler (Page 2373)

10.7.3 Basics

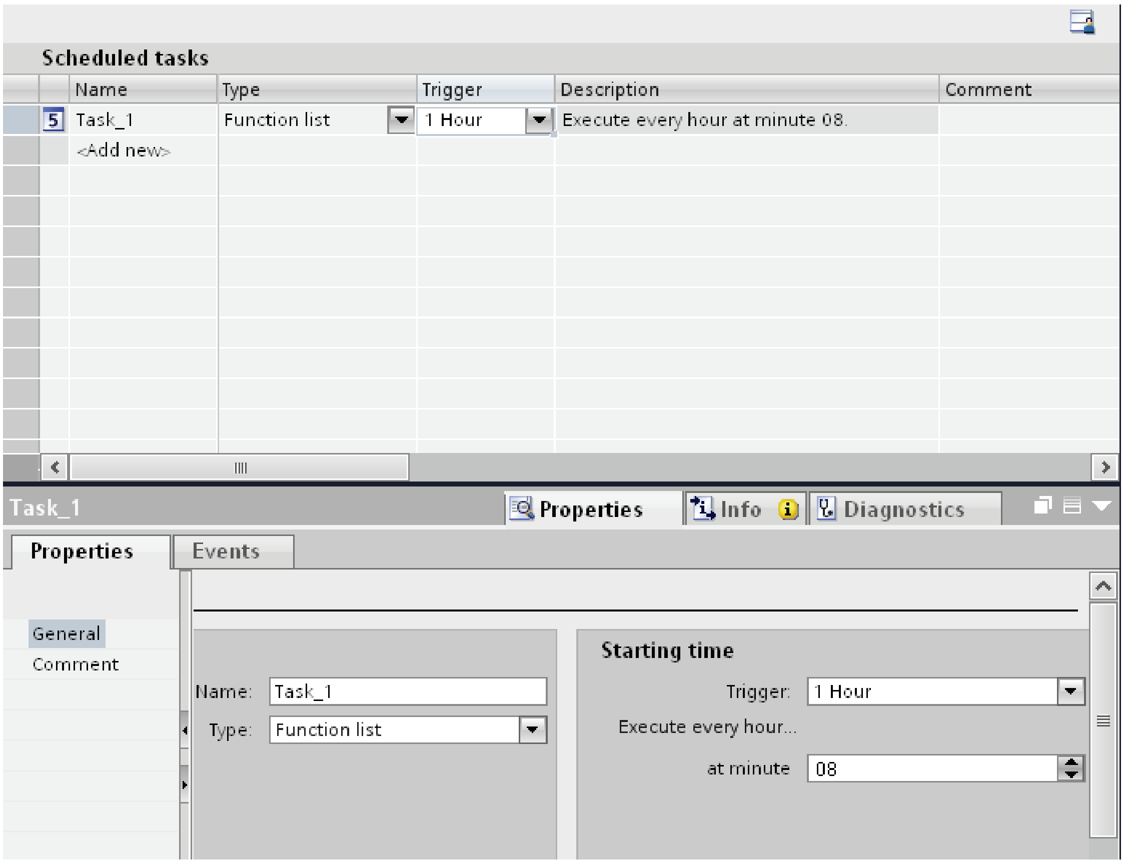
10.7.3.1 Work area of the "Scheduler" editor

Introduction

Double-click on "Scheduler" to open it in the project view. The work area shows the scheduled tasks, which consist of the trigger and the task type, for example, the function list.

Structure

The work area consists of the table of jobs.



Scheduled tasks

Name	Type	Trigger	Description	Comment
Task_1	Function list	1 Hour	Execute every hour at minute 08.	
<Add new>				

Task_1

General

Name: Task_1

Type: Function list

Starting time

Trigger: 1 Hour

Execute every hour...

at minute 08

The table of tasks shows specified tasks with their properties, such as triggers. You select a task type and a trigger. You assign a name and a comment to the task. The description provides a written summary of the task including the timing for the task.

Inspector window

The "Properties" tab of the Inspector window is split into two parts.

The "Job" area lists the name of the job and the job type. The "Starting time" area shows the trigger. The area is different depending on the trigger selected.

In the "Events" tab use the function list with system functions that will be executed in the task.

Note

You can obtain more detailed information about the elements of the user interface using the tooltips. To do so, move the mouse pointer to the relevant object or press <F1> if the object has already been selected.

See also

Field of application of the Scheduler (Page 2373)

Planning tasks with event triggers (Page 2377)

Triggers (Page 2377)

Function list (Page 2376)

10.7.3.2 Function list

Function list

A trigger starts the function list. The function list is executed line-for-line. Each line contains a system function. You can configure exactly one function list for each task.

Note

The choice of configurable system functions in a function list depends on the selected trigger and the HMI device.

See also

Work area of the "Scheduler" editor (Page 2375)

10.7.3.3 Triggers

Introduction

A trigger is linked to a task and forms the triggering event which will call this task. The task is executed when the trigger occurs.

Event trigger

When a task is linked to a system event, the task will be triggered by the event. System events include, for example, Runtime stop, screen change, user change, etc.

Each system event can only be configured once for each HMI device.

Deactivating job

If you do not need a certain job temporarily, deactivate the job in the Engineering System. You also use the trigger "Deactivated" to make a previously configured system event available once again.

Example: Task "A" is planned with the system event "Shutdown". This system event is then no longer available for another task "B". Select "Disabled" as the trigger for task "A" to make the "Runtime stop" system event available again.

Note

The available triggers depend on the HMI device.

See also

Work area of the "Scheduler" editor (Page 2375)

10.7.3.4 Planning tasks with event triggers

Introduction

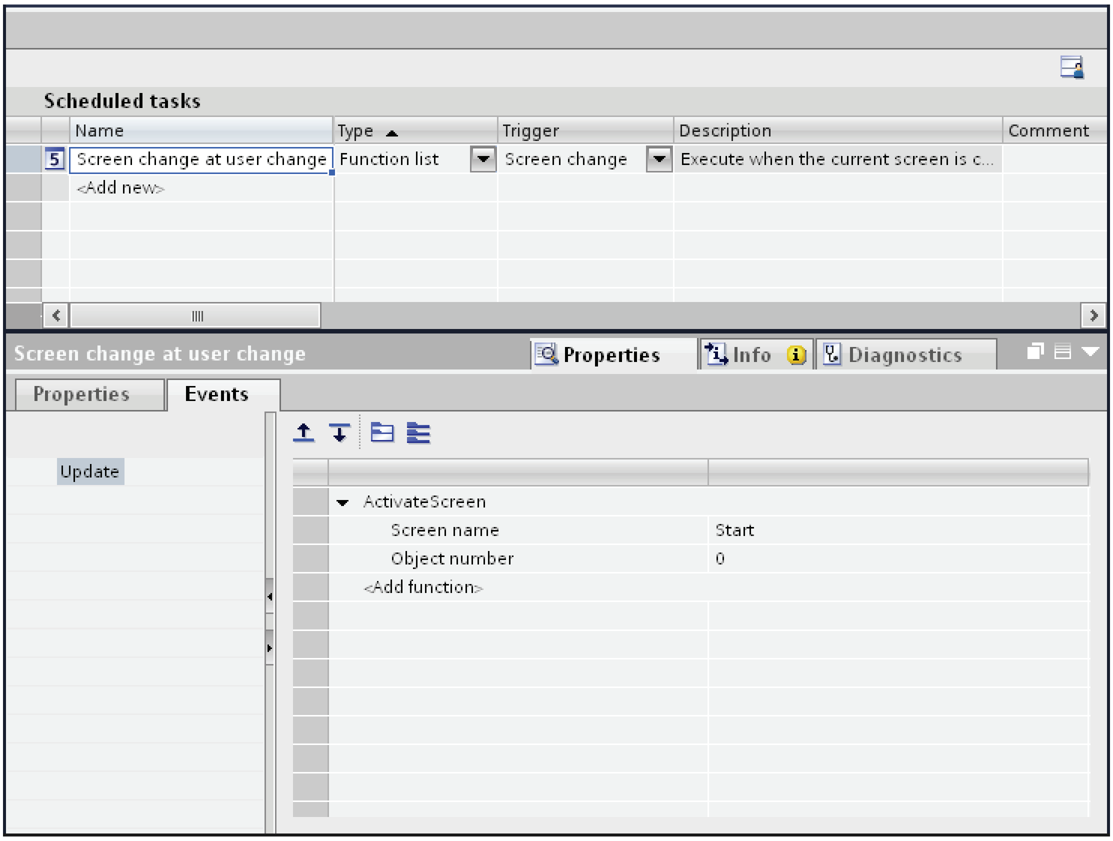
You plan a task that generates a screen change when the user changes.

Requirements

- The "Scheduler" work area is open.
- You have created the "Start" screen.

Procedure

1. Click "Add..." in the table of the task area.
2. Enter "Screen change at user change" as the "Name."
3. Select "User change" as the "Trigger."
4. In the Inspector window, select "Properties > Events".
5. Select the "Screen/ActivateScreen" system function in the function list.
6. Select the "Start" screen in the screen name field.



Scheduled tasks

	Name	Type	Trigger	Description	Comment
5	Screen change at user change	Function list	Screen change	Execute when the current screen is c...	
	<Add new>				

Screen change at user change

ActivateScreen	
Screen name	Start
Object number	0
<Add function>	

Result

The task is executed with the "User change" event. When a new user logs on successfully, the "Start" screen is called up.

See also

Work area of the "Scheduler" editor (Page 2375)

10.7.4 Examples

10.7.4.1 Example: Update user following change of user

Task

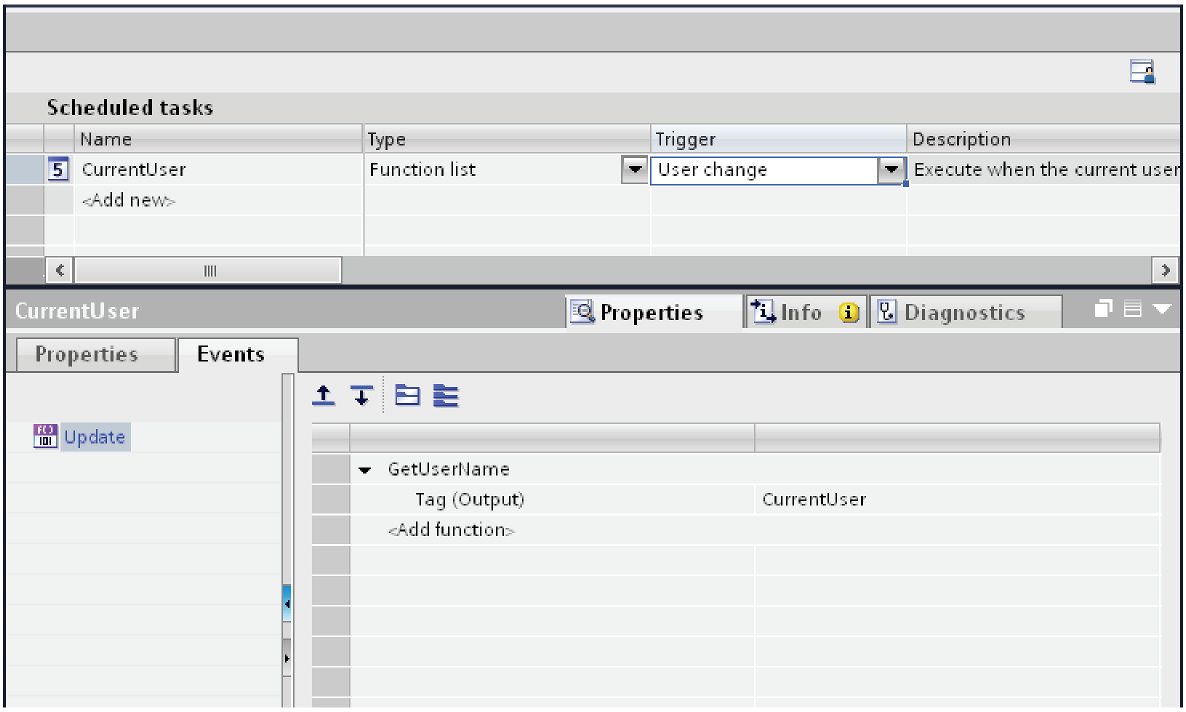
Configure an I/O field which displays the logged on user. Configure a task which updates the I/O field when the logged on user changes.

Requirements

- A "CurrentUser" tag of the "String" type is created.
- A screen has been created and opened.
- An I/O field is created in the screen.

Procedure

1. Click on the "I/O field" object.
2. In the Inspector window, select "Properties > Events > General":
   - Select "Character string" as the "Display format."
   - Select "CurrentUser" as the "Variable."
   - Select "Output" as the mode.
3. Change to the work area of the Scheduler.
4. Click "Add..." in the table of the task area.
5. Enter "CurrentUser" as the "Name".
6. Select "User change" as the "Trigger."
7. In the Inspector window, select "Properties > Events".
8. Select the system function "ReadUserName" from the "User Management" group in the function list.
9. Select "CurrentUser" as the "Variable."



Scheduled tasks

	Name	Type	Trigger	Description
5	CurrentUser	Function list	User change	Execute when the current user
	<Add new>			

CurrentUser

GetUserName	
Tag (Output)	CurrentUser
<Add function>	

Result

When a new user logs on successfully, the "ReadUserName" function is called up. The "CurrentUser" tag is updated and displayed in the I/O field of the newly logged on user.

If a user does not log on successfully, the logged on user is logged off. The I/O field continues to display the user previously logged on until a new user logs on successfully.

See also

Field of application of the Scheduler (Page 2373)

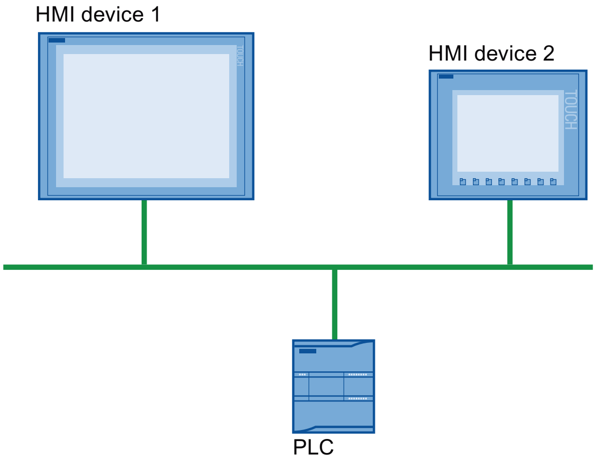
10.8 Communicating with PLCs

10.8.1 Basics of communication

10.8.1.1 Communication between devices

Communication

The data exchange between two devices is known as communication. The devices can be interconnected directly or via a network. The interconnected devices in communication are referred to as communication partners.



Data transferred between the communication partners may serve different purposes:

- Display processes
- Operate processes
- Output alarms
- Archive process values and alarms
- Document process values and alarms
- Administer process parameters and machine parameters

Communication partners

Communication between the following devices is described in more detail in this section:

- PLC

  The PLC controls a process by means of a user program.

- HMI device

  You use the HMI device to operate and monitor the process.

Basic information for all communication

The basis for all types of communication is a network configuration. In a network configuration, you specify the connection that exists between the configured devices.

With the network configuration, you also ensure the necessary prerequisites for communication, in other words:

- Every device in a network is assigned a unique address.
- The devices carry out communication with consistent transmission characteristics.

Automation system

The following characteristics describe an automation system:

- The PLC and HMI device are interconnected
- The network between the PLC and HMI device is configured

Communication between HMI devices

The HTTP protocol is available for communication between HMI devices.

For more detailed information, refer to the documentation on the SIMATIC HMI HTTP protocol.

Communication via a uniform and vendor-neutral interface

With OPC (Openess Productivity Collaboration), WinCC has a uniform and manufacturer-neutral software interface. This interface enables standardized data exchange between industrial, office, and manufacturing applications.

For more detailed information, refer to the documentation for OPC.

10.8.1.2 Devices and networks in the automation system

Introduction

To set up an automation system, you must configure, parameterize, and interconnect the individual devices.

You insert PLCs and HMI devices into the project in the same way. Likewise, you configure the two devices in the same way.

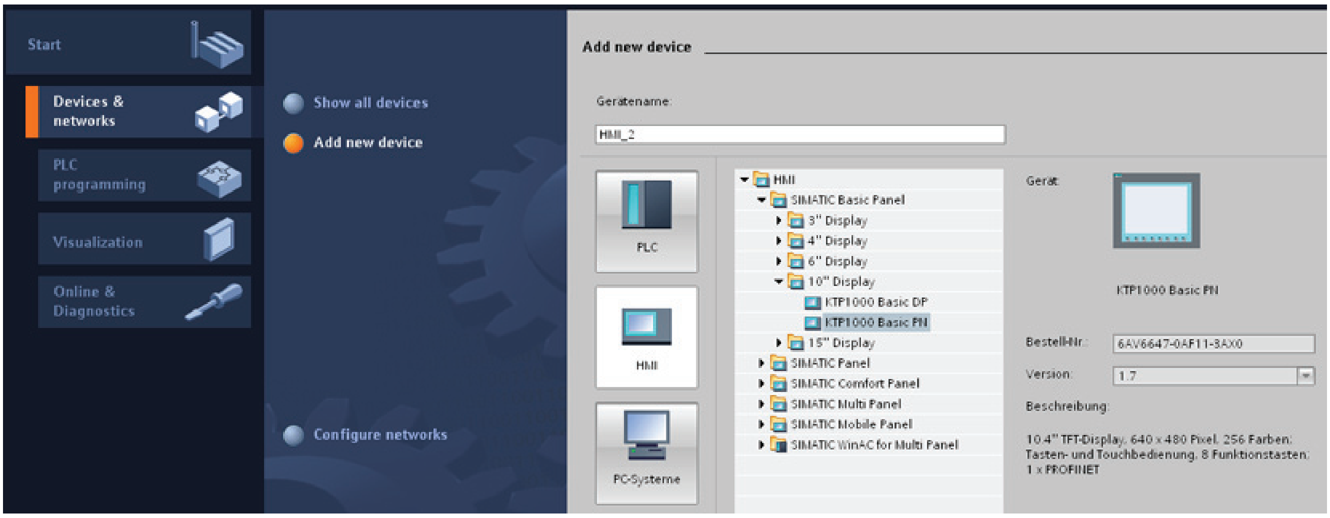
Automation system setup:

1. Insert PLC into the project.
2. Insert HMI device into the project.
3. Network the devices together.
4. Interconnect the devices.

Inserting devices

If you have created a project, you can add a device in the portal view or project view.

- Portal view



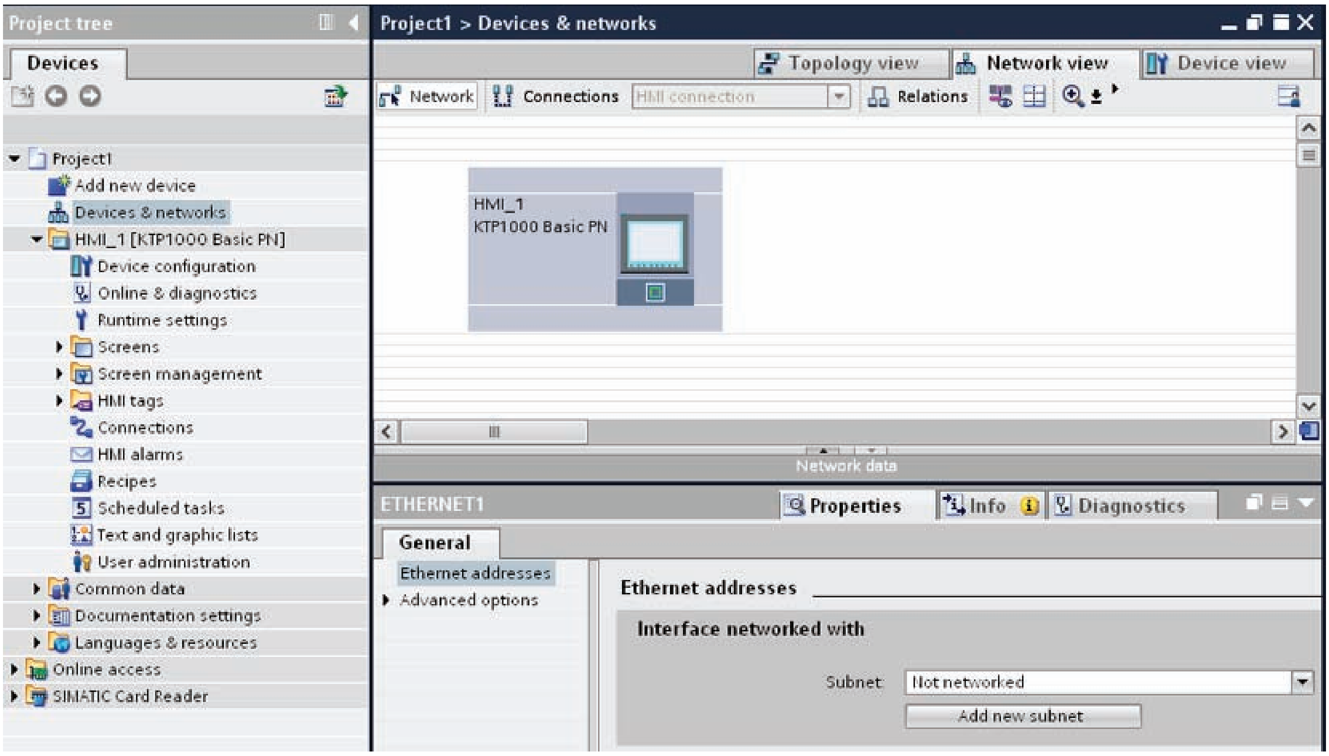
Gerätename: HMI_2

Bestell-Nr.: 6AV6647-0AF11-3AX0

Version: 1.7

Beschreibung: 10.4" TFT-Display, 640 x 480 Pixel, 256 Farben; Tasten- und Touchbedienung, 8 Funktionstasten; 1 x PROFINET

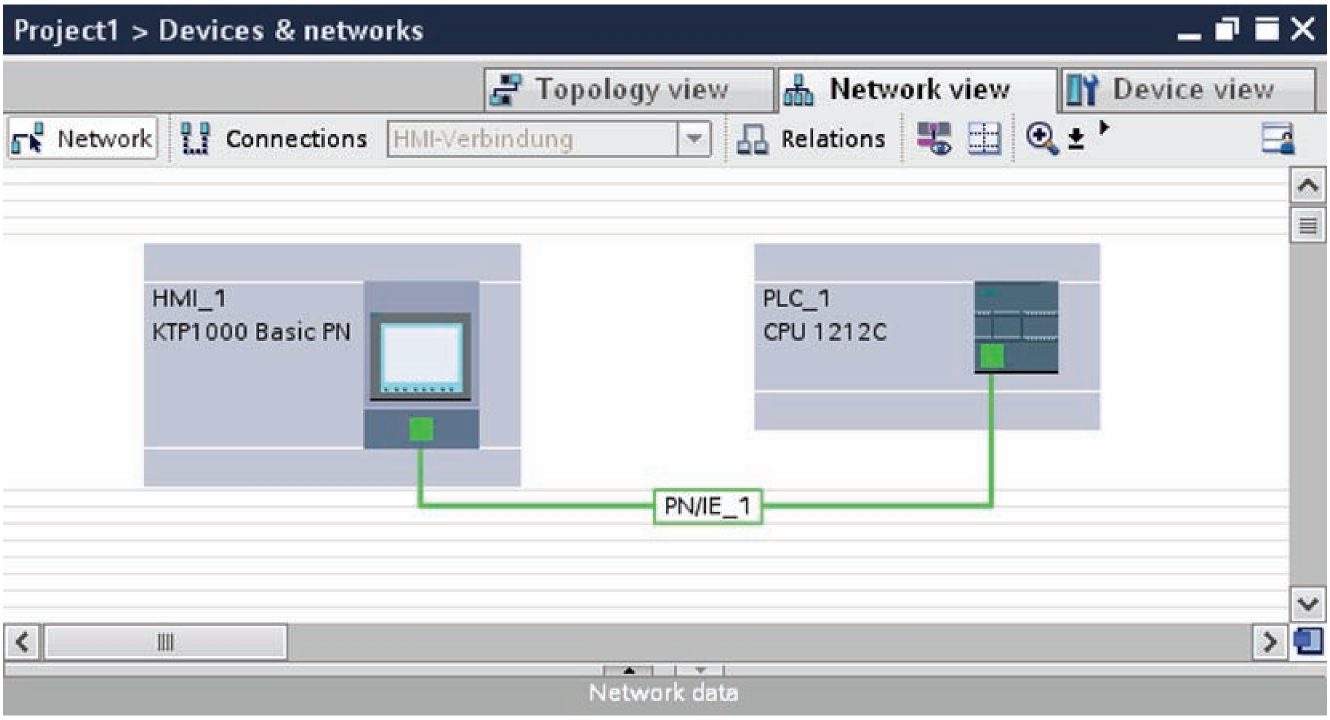
- Project view



Subnet: Not networked

Networking devices

You can network the interfaces of the communication-capable devices conveniently in the "Devices & Networks" editor. In the networking step, you configure the physical device connections.

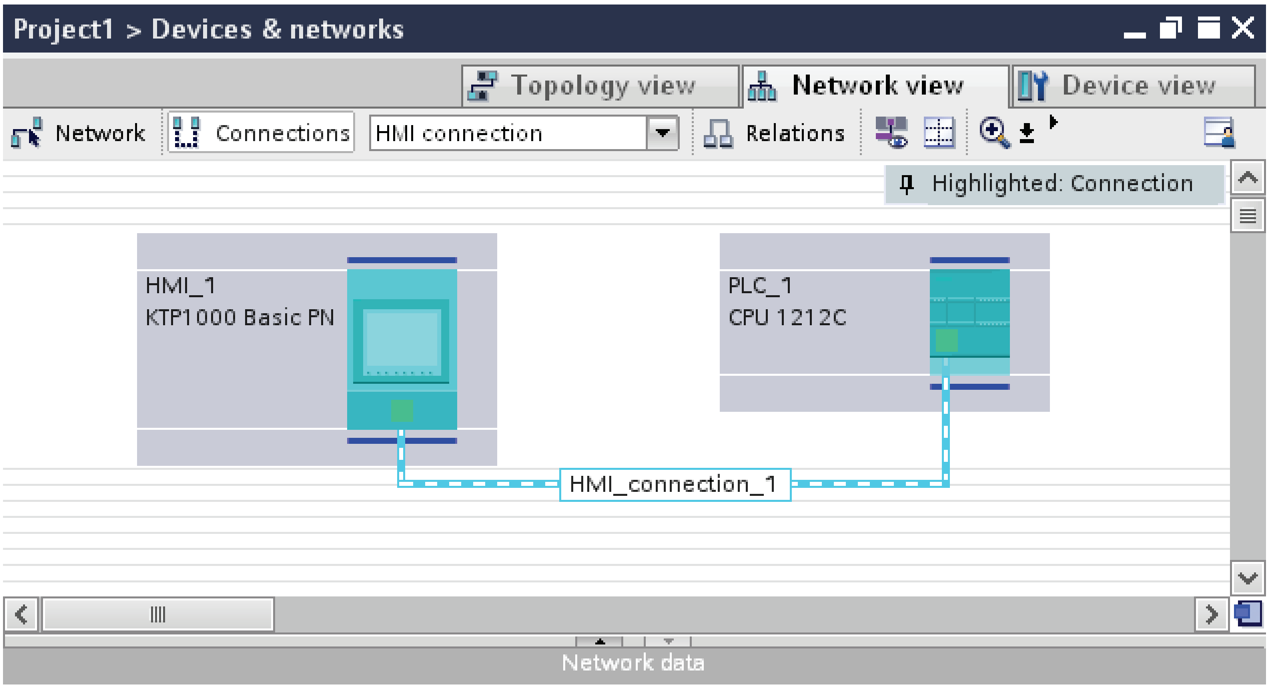


The tabular network overview supplements the graphical network view with the following additional functions:

- You obtain detailed information on the structure and parameter settings of the devices.
- Using the "Subnet" column, you can connect communication-capable components to subnets that have been created.

Connecting devices

After you network the devices together, you configure the connection. You configure the "HMI connection" connection type for communication with the HMI device.



10.8.1.3 Data exchange using tags

Communication using tags

Process values are forwarded in runtime using tags. Process values are data which is stored in the memory of one of the connected automation systems. They represent the status of a plant in the form of temperatures, fill levels or switching states, for example. Define external tags for processing the process values in WinCC.

WinCC works with two types of tag:

- External tags
- Internal tags

Working with tags

See the chapter "Working with tags (Page 2122)" for further information about configuring tags.

10.8.1.4 Data exchange using area pointers

Communication using area pointers

Area pointers are parameter fields. WinCC receives information about the location and size of data areas in the PLC from these parameter fields in runtime.

During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

Based on the evaluation of data stored in the data areas, the PLC and HMI device initiate mutually defined actions.

The area pointers are managed centrally in the "Connections" editor. Area pointers are used to exchange data from specific user data areas.

You use the following area pointers in WinCC:

- Data record
- Date/time
- Coordination
- Job mailbox
- Date/time PLC
- Project ID
- Screen number

The availability of the various area pointers is determined by the HMI device used.

10.8.1.5 Communication drivers

Communication drivers

A communication driver is a software component that establishes a connection between a PLC and an HMI device. The communication driver thus enables the assignment of process values to HMI tags.

The interface as well as the profile and transmission speed can be chosen, depending on the HMI device used and the connected communication partner.

10.8.2 Networks and connections

10.8.2.1 SIMATIC communication networks

Communication networks

Overview

Communication networks are a central component of modern automation solutions. Industrial networks have to fulfill special requirements, for example:

- Coupling of automation systems as well as simple sensors, actuators, and computers.
- The information has to be correct and has to be transferred at the right moment.
- Robust against electromagnetic disturbances, mechanical stresses and soiling
- Flexible adaptation to the production requirements

Industrial networks belong to the LANs (Local Area Networks) and allow communication within a limited area.

Industrial networks fulfill the following communication functions:

- Process and field communication of the automation systems including sensors and actuators
- Data communication between automation systems
- IT communication for integrating the modern information technology

Overview of the networks

This section examines the following networks:

- Industrial Ethernet

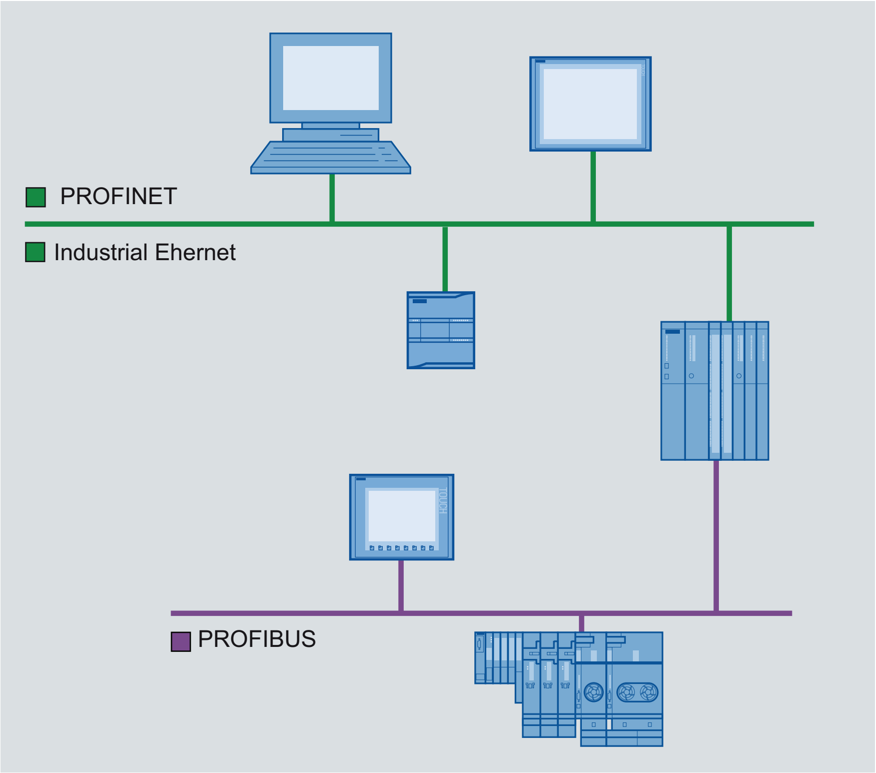
  The industrial network standard for all levels
- PROFINET

  The open Industrial Ethernet standard for automation
- PROFIBUS

  The international standard for the field area and market leader at the field busses
- MPI

  The integrated interface of the SIMATIC products
- PPI

  The integrated interface specially for the S7-200



PROFINET and Ethernet

Industrial Ethernet

Industrial Ethernet, which is based on IEEE 802.3, enables you to connect your automation system to your office networks. Industrial Ethernet provides IT services that you can use to access production data from the office environment.

Ethernet network

An Ethernet network allows you to interconnect all devices that are connected to the network via an integrated Ethernet interface or a communication module. You can thereby connect multiple HMI devices to one SIMATIC S7 PLC and multiple SIMATIC S7 PLCs to one HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used. Additional information is available in the documentation for the respective HMI device.

PROFINET

PROFINET is an open standard for industrial automation defined by IEEE 61158 and based on Industrial Ethernet. PROFINET makes use of IT standards all the way to the field level and enables plant-wide engineering.

With PROFINET, you can realize high-performance automation solutions for applications with stringent real-time requirements.

PROFIBUS

PROFIBUS DP

PROFIBUS DP (distributed I/O) is used to connect the following devices:

- PLCs, PCs, HMI devices
- Distributed I/O devices, e.g., SIMATIC ET 200
- Valves
- Drives

PROFIBUS DP's fast response times make it ideally suited for the manufacturing industry.

Its basic functionality includes cyclic exchange of process data between the master and PROFIBUS DP slaves, as well as diagnostics.

PROFIBUS network

You can connect an HMI device within the PROFIBUS network to any SIMATIC S7 module that has an integrated PROFIBUS or PROFIBUS DP interface. You can thereby connect multiple HMI devices to one SIMATIC S7 PLC and multiple SIMATIC S7 PLCs to one HMI device.

The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used. Additional information is available in the documentation for the respective HMI device.

You configure the SIMATIC S7-200 PLC as a passive device in the network. You connect the SIMATIC S7-200 using the DP connector or a PROFIBUS communication module.

MPI

MPI

MPI (Multi-Point Interface) is the integrated interface for SIMATIC products:

- PLCs
- HMI devices
- Programming device/PC

Small subnets with the following characteristics are set up with MPI:

- Short distances
- Few devices
- Small data quantities

MPI network

You connect the HMI device to the MPI interface of the SIMATIC S7 PLC. You can connect multiple HMI devices to one SIMATIC S7 PLC and multiple SIMATIC S7 PLCs to one HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used. Additional information is available in the documentation for the respective HMI device.

Network architectures

MPI is based on the PROFIBUS standard (IEC 61158 and EN 50170) and supports the following bus topologies:

- Line
- Star
- Tree

An MPI subnet contains a maximum of 127 devices and consists of multiple segments. Each segment contains a maximum of 32 devices and is limited by terminating resistors. Repeaters are used to connect segments. The maximum cable length without a repeater is 50 m.

PPI

Introduction

PPI (point-to-point interface) is an integrated interface that was developed specially for the SIMATIC S7-200. A PPI network typically connects S7-200 PLCs. However, other SIMATIC PLCs (e.g., S7-300 and S7-400) or HMI devices can communicate with a SIMATIC S7-200 in the PPI network.

PPI network

A PPI connection is a point-to-point connection. The HMI device is the master. The SIMATIC S7-200 is the slave.

You can connect a maximum of one SIMATIC S7-200 to an HMI device. You use the serial connector of the CPU to connect the HMI device. You can connect multiple HMI devices to one SIMATIC S7-200. From the perspective of the SIMATIC S7-200, only one connection at a time is possible.

Note

The PPI network can contain a maximum of four masters in addition to the HMI device. For performance reasons, do not configure more than four devices at a time as a master in the PPI network.

Network architectures

PPI is based on the PROFIBUS standard (IEC 61158 and EN 50170) and supports the following bus topologies:

- Line
- Star

Multi-master networks with a maximum of 32 masters are set up with PPI:

- An unlimited number of masters can communicate with each slave.
- A slave can be assigned to multiple masters.

The RS 485 repeater can be used to extend the PPI network. Modems can also be connected to the PPI network.

10.8.2.2 Configuring networks and connections

Networking devices

Introduction

The "Devices & Networks" editor is provided for configuring connections. You can network devices in the editor. You can also configure and assign parameters to devices and interfaces. You then configure the required connections between the networked devices.

In the "Devices & Networks" editor you configure HMI connections with the PLCs:

- SIMATIC S7 1200
- SIMATIC S7 300
- SIMATIC S7 400

You configure the HMI connections to other PLCs in the "Connections" editor of the respective HMI device.

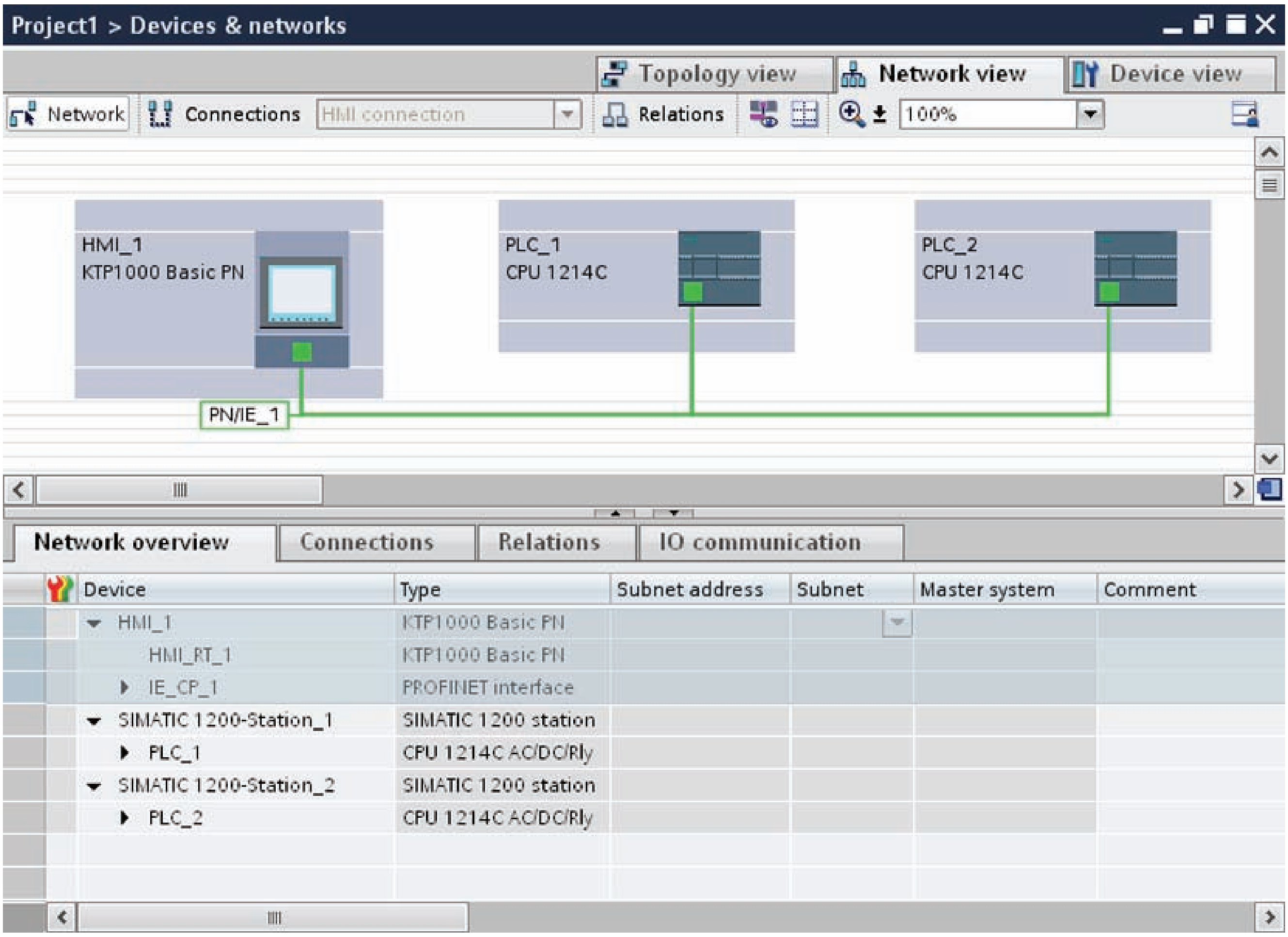
Networking devices

The network view of the "Devices & Networks" editor includes a graphical area and a tabular area. You can use the graphical area to network the devices in the project with drag-and-drop. The tabular area provides an overview of the devices and their components.

You can network the following PLCs together with HMI devices in the "Devices & Networks" area.

- SIMATIC S7 1200
- SIMATIC S7 300
- SIMATIC S7 400

All other PLCs are available in the TIA-Portal and are configured "not integrated". You configure "not integrated" connections in the "Connections" editor of the HMI device.



Device	Type	Subnet address	Subnet	Master system	Comment
HMI_1	KTP1000 Basic PN				
HMI_RT_1	KTP1000 Basic PN				
IE_CP_1	PROFINET interface				
SIMATIC 1200-Station_1	SIMATIC 1200 station				
PLC_1	CPU 1214C AC/DC/Rly				
SIMATIC 1200-Station_2	SIMATIC 1200 station				
PLC_2	CPU 1214C AC/DC/Rly				

With the networking step, you configure the physical connection of the communication partners. The networking of devices is depicted by lines that are colored according to the interface.

Configuring an integrated connection in the "Devices & Networks" editor

Introduction

You configure an HMI connection between the HMI device and a SIMATIC S7 1200 in the "Devices & Networks" editor. This HMI connection is the direct connection between the communication partners that you have created in a project.

Integrated connections

Connections of devices within a project are referred to as integrated connections. In the case of integrated connections, you can directly configure addresses of PLC tags.

Note

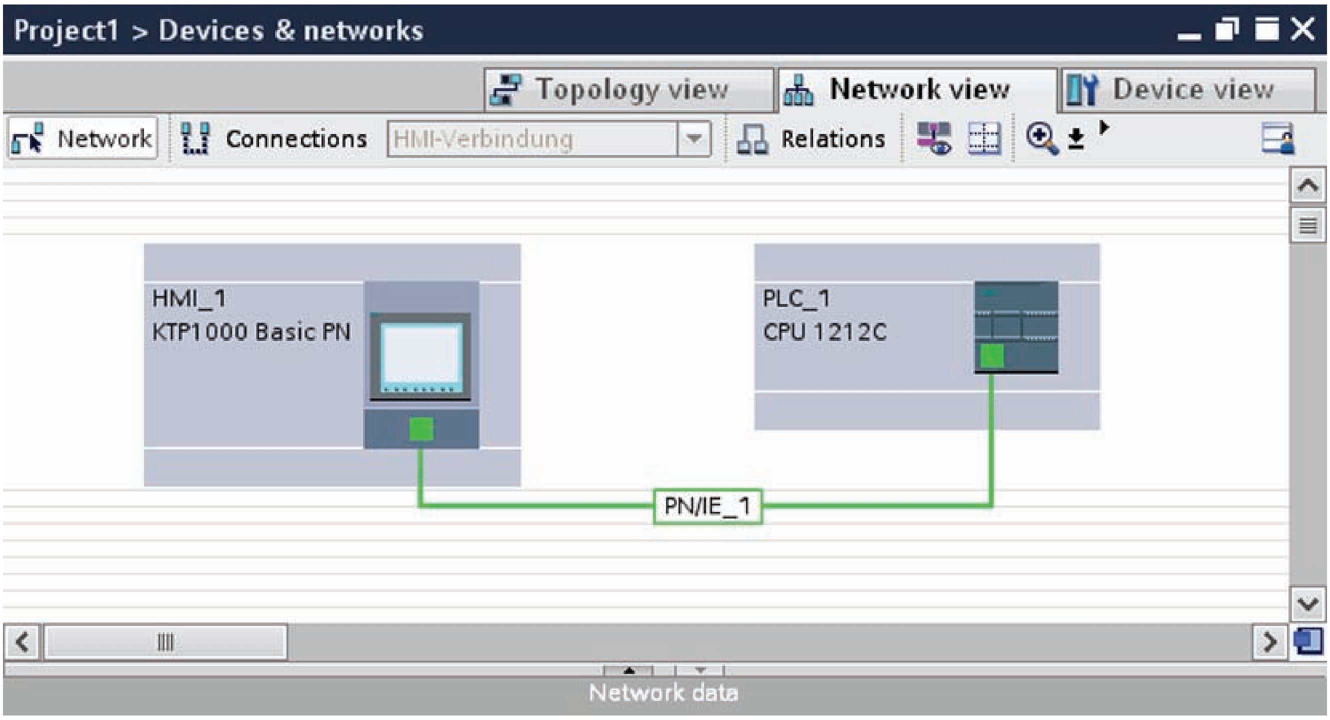
An HMI connection can be configured in the "Devices & Networks" editor for the following PLCs only:

- SIMATIC S7 1200
- SIMATIC S7 300
- SIMATIC S7 400

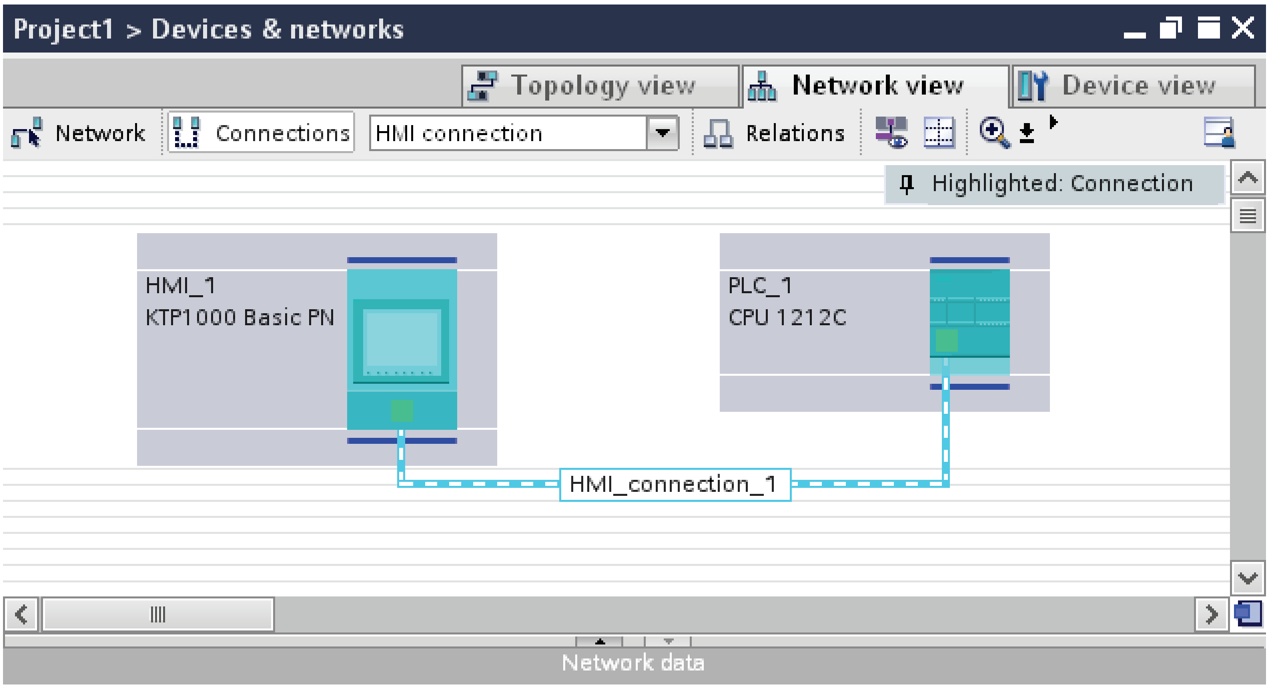
You configure the HMI connections to all other PLCs in the "Connections" editor of the HMI device.

Configuring an HMI connection in the "Devices & Networks" editor

1. Insert an HMI device and a SIMATIC S7 1200 into your project.



2. Switch to "Connections" mode.
3. Select the "HMI connection" connection type.
4. Use a drag-and-drop operation to interconnect the two PROFINET interfaces.



5. Change the IP address and subnet mask address parameters according to the requirements of your project.

Special considerations of the "Devices & Networks" editor

Introduction

If you are configuring or have already configured networks or HMI connections, the "Devices & Networks" editor supports you with the following functions:

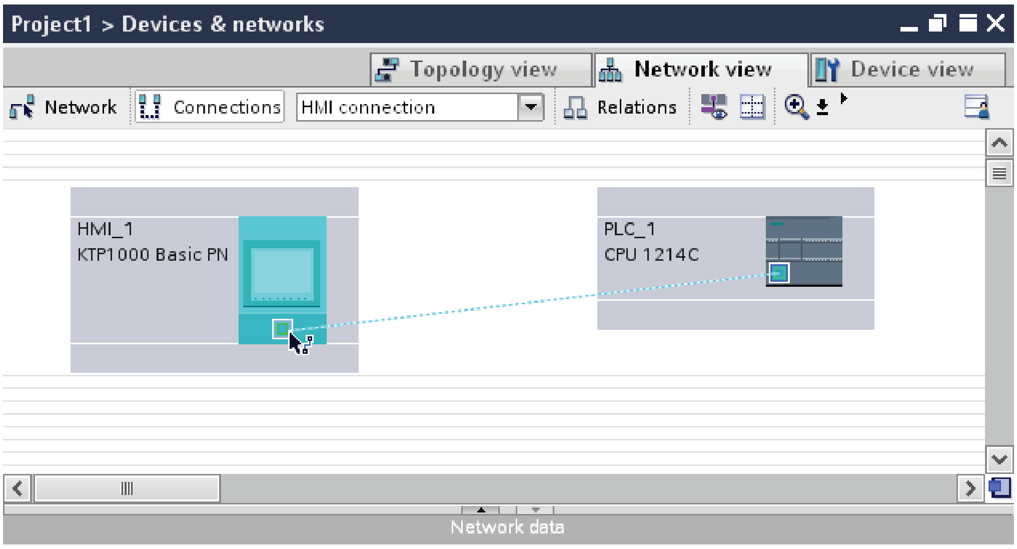
- Highlighting of communication partners
- Highlighting of HMI connections
- Automatic creation of subnets

Highlighting of communication partners

All communication partners for which an HMI connection is possible are highlighted in turquoise if you have selected the "HMI connection" type.

Starting from the interface of a device create an HMI connection to the device of another device using a drag-and-drop operation. During the drag-and-drop operation all potential communication partners are highlighted in turquoise.

Use the ESC key to stop connecting interfaces using a drag-and-drop operation.



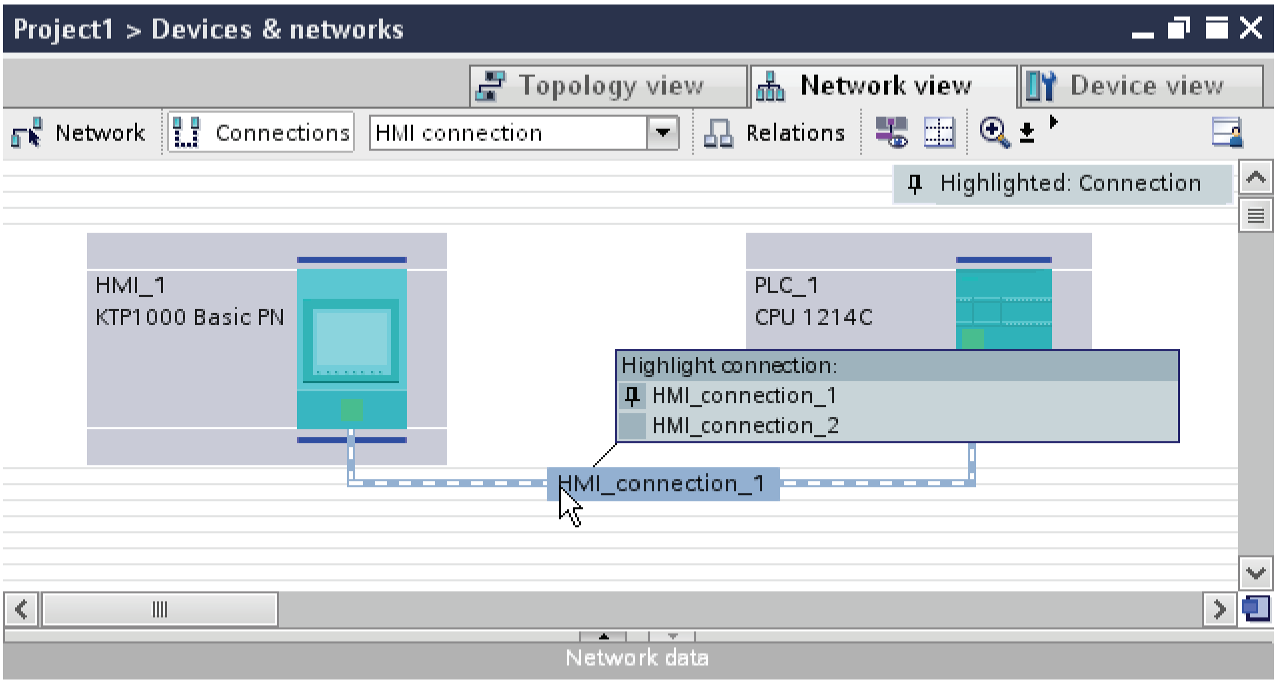
When the mouse pointer is moved over the interface of a device, the following icons indicate whether a connection is possible:

A connection is possible.

A connection is not possible.

Highlighting of HMI connections

A turquoise highlighting of the connection indicates that a HMI connection was created. If several HMI connections are created, you can select one of the already created HMI connections in a dialog.



Then you can configure the parameters of the selected HMI connection and the communication partners in the inspector window.

Automatic creation of subnets

A subnet is created automatically when you configure a HMI connection.

Configuring a non-integrated connection in the "Connections" editor

Introduction

You use the "Connections" editor of the HMI device to configure a connection between an HMI device and a PLC that cannot be configured in the "Devices & Networks" editor.

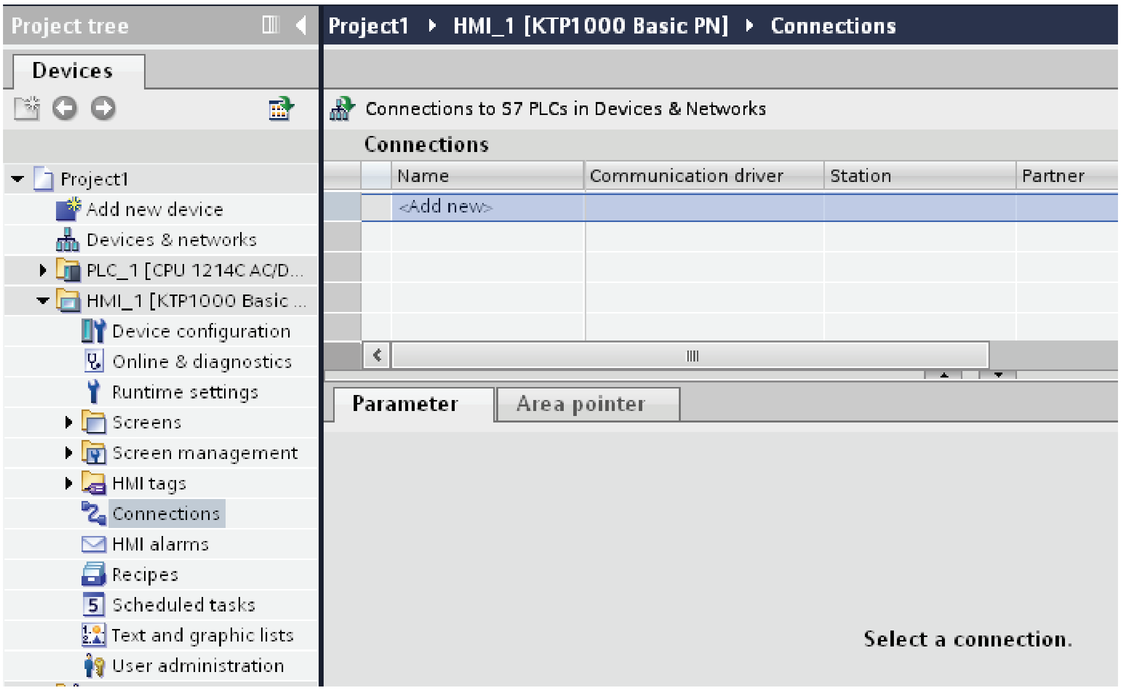
These connections are referred to as non-integrated connections.

Requirements

- A project is open.
- An HMI device has been created.

Configuring a connection in the "Connections" editor

1. Open the "Connections" editor of the HMI device.
2. Create a new connection.

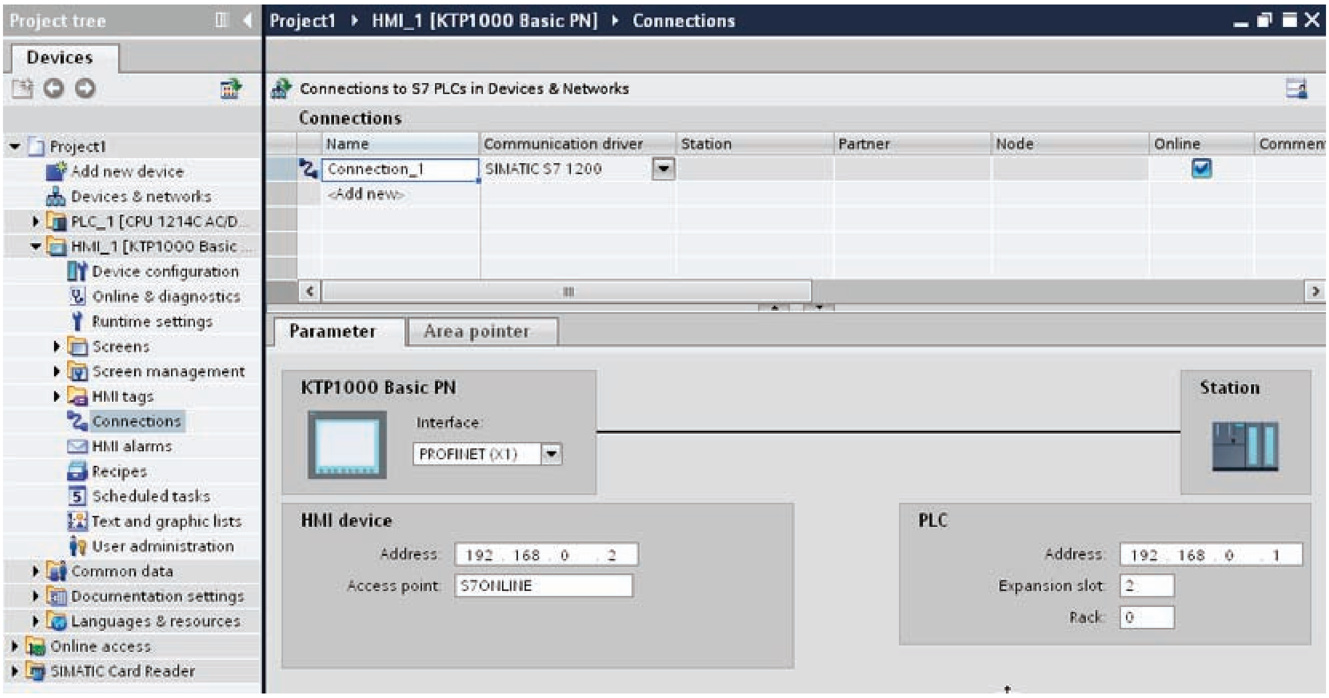


Connections

Name	Communication driver	Station	Partner
<Add new>			

Select a connection.

3. Select the communication driver.
4. Set the connection parameters.



Connections

Name	Communication driver	Station	Partner	Node	Online	Commen
Connection_1	SIMATIC S7 1200				☑	
<Add new>						

KTP1000 Basic PN

Interface: PROFINET (X1)

HMI device

Address: 192 . 168 . 0 . 2

Access point: S7ONLINE

PLC

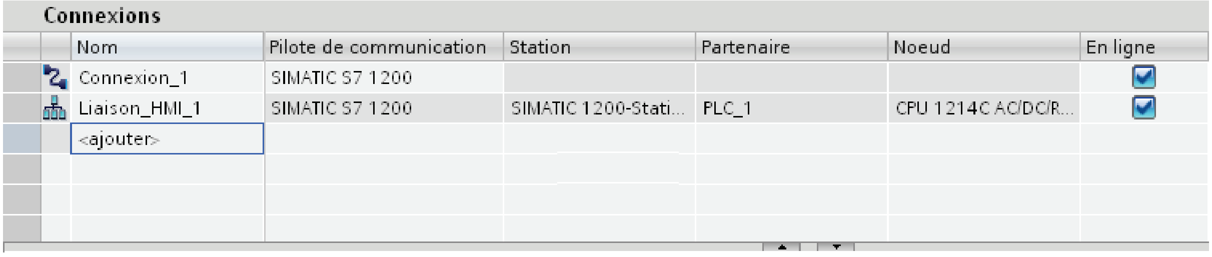
Address: 192 . 168 . 0 . 1

Expansion slot: 2

Rack: 0

Integrated connections in the "Connections" editor

If you have already configured the integrated connections of the HMI device in the "Devices & Networks" editor, they are also displayed in the "Connections" editor.



Connexions

Nom	Pilote de communication	Station	Partenaire	Noeud	En ligne
Connexion_1	SIMATIC S7 1200				☑
Liaison_HMI_1	SIMATIC S7 1200	SIMATIC 1200-Stati...	PLC_1	CPU 1214C AC/DC/R...	☑
<ajouter>					

Meaning of the icons used:

Integrated connection

Non-integrated connection

10.8.3 Data exchange

10.8.3.1 Data exchange using tags

Basics of tags

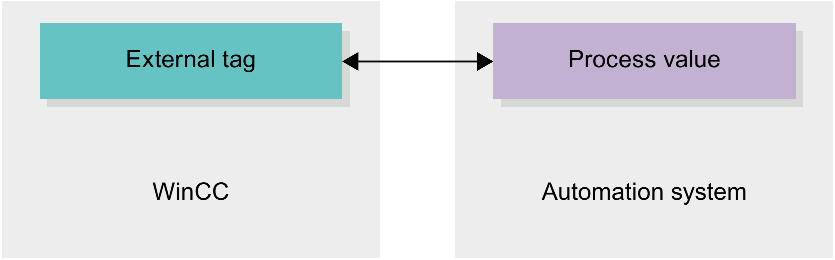
Introduction

Process values are forwarded in runtime using tags. Process values are data which is stored in the memory of one of the connected automation systems. They represent the status of a plant in the form of temperatures, fill levels or switching states, for example. Define external tags for processing the process values in WinCC.

WinCC works with two types of tag:

- External tags
- Internal tags

The external tags form the link between WinCC and the automation systems. The values of external tags correspond to the process values from the memory of an automation system. The value of an external tag is determined by reading the process value from the memory of the automation system. It is also possible to rewrite a process value in the memory of the automation system.



Internal tags do not have a process link and only convey values within the WinCC.

Tags in WinCC

For external tags, the properties of the tag are used to define the connection that the WinCC uses to communicate with the automation system and form of data exchange.

Tags that are not supplied with values by the process - the internal tags - are not connected to the automation system. In the tag's "Connection" property, this is identified by the "Internal tag" entry.

You can create tags in different tag tables for greater clarity. You then directly access the individual tag tables in the "HMI tags" node in the project tree. The tags from all tag tables can be displayed with the help of the table "Show all tags".

With structures you bundle a number of different tags that form one logical unit. Structures are project-associated data and are available for all HMI devices of the project. You use the "Types" editor in the project library to create and edit a structure.

Overview of HMI tag tables

Introduction

HMI tag tables contain the definitions of the HMI tags that apply across all devices. A tag table is created automatically for each HMI device created in the project.

In the project tree there is an "HMI tags" folder for each HMI device. The following tables can be contained in this folder:

- Default tag table
- User-defined tag tables
- Table of all tags

In the project tree you can create additional tag tables in the HMI tags folder and use these to sort and group tags and constants. You can move tags to a different tag table using a drag-and-drop operation or with the help of the "Tag table" field. Activate the "Tags table" field using the shortcut menu of the column headings.

Default tag table

There is one default tag table for each HMI device of the project. It cannot be deleted or moved. The default tag table contains HMI tags and, depending on the HMI device, also system tags. You can declare all HMI tags in the standard tags table or, as necessary, additional user-defined tables of tags.

User-defined tag tables

You can create multiple user-defined tag tables for each HMI device in order to group tags according to your requirements. You can rename, gather into groups, or delete user-defined tag tables. To group tag tables, create additional subfolders in the HMI tags folder.

All tags

The "All tags" table shows an overview of all HMI tags and system tags of the HMI device in question. This table cannot be deleted, renamed or moved. This table also contains the "Tags table" column, which indicates the tag table of where a tag is included. Using the "Tags table" field, the assignment of a tag to a tags table can be changed.

With devices for Runtime Professional, the table "All tags" contain an additional tab "System tags". The system tags are created by the system and used for internal management of the project. The names of the system tags begin with the "@" character. System tags cannot be deleted or renamed. You can evaluate the value of a system tag, but cannot modify it.

Additional tables

The following tables are also available in an HMI tag table:

- Discrete alarms
- Analog alarms
- Logging tags

With the help of these tables you configure alarms and logging tags for the currently selected HMI tag.

Discrete alarms table

In the "Discrete alarms" table, you configure discrete alarms to the HMI tag selected in the HMI tag table. When you configure a discrete alarm, multiple selection in the HMI tag table is not possible. You configure the discrete alarms for each HMI tag separately.

Analog alarms table

In the "Analog alarms" table, you configure analog alarms to the HMI tag selected in the HMI tag table. When you configure an analog alarm, multiple selection in the HMI tag table is not possible. You configure the analog alarms for each HMI tag separately.

Logging tags table

In the "Logging tags" table, you configure logging tags to the HMI tag selected in the HMI tag table. When you configure a logging tag, multiple selection in the HMI tag table is not possible. You configure the logging tags for each HMI tag separately. The "Logging tags" table is only available if the HMI device used supports logging.
If WinCC Runtime Professional is used, you can also assign several log tags to a tag. With the other HMI devices, you can only assign one log tag to a tag.

External tags

Introduction

External tags allow communication (exchange of data) between the components of an automation system, such as between the HMI device and the PLC.

Principle

An external tag is the image of a defined memory location in the PLC. You have read and write access to this storage location from both the HMI device and from the PLC.

Since external tags are the image of a storage location in the PLC, the applicable data types depend on the PLC which is connected to the HMI device.

In STEP 7, if you write a PLC control program, the PLC tags created in the control program will be added to the PLC tag table. If you want to connect an external tag to a PLC tag, access the PLC tags directly via the PLC tag table and connect them to the external tag.

Data types

All the data types which are available at the connected PLC are available at an external tag in WinCC. Information about data types which are available for connection to other PLCs can be found in the documentation about the respective communication drivers.

See "Communication between devices (Page 2381)" for additional information.

Note

As well as external tags, area pointers are also available for communication between the HMI device and PLC. You can set up and enable the area indicators in the "Connections" editor.

Update of tag values

For external tags, the current tag values are transmitted in runtime via the communication connection between WinCC and the connected automation systems and then saved in the runtime memory. Next, the tag value will be updated to the set cycle time. For use in the runtime project, WinCC accesses tag values in the runtime memory that were read from the PLC at the previous cycle time. As a result, the value in the PLC can already change whilst the value from the runtime memory is being processed.

See also

Communication between devices (Page 2381)

Addressing external tags

Introduction

The options for addressing external tags depend on the type of connection between WinCC and the PLC in question. A distinction must be made between the following connection types:

- Integrated connection
  Connections of devices which are within a project and were created with "Devices & Networks" editor are referred to as integrated connections.
- Non-integrated connection
  Connections of devices which were created with the "Connections" editor are referred to as non-integrated connections. It is not necessary that all of the devices be within a single project.

The connection type can also be recognized by its symbol.

🖧	Integrated connection
⤵	Non-integrated connection

You can find additional details on this in the section "Basics of communication (Page 2381)".

Addressing with integrated connections

An integrated connection offers the advantage that you can address a tag both symbolically and absolutely.

For symbolic addressing, you select the PLC tag via its name and connect it to the HMI tag. The valid data type for the HMI tag is automatically selected by the system. You have to distinguish between the following cases when you address elements in data blocks:

- Symbolic addressing of data blocks with optimized access
  For symbolic addressing of a data block with optimized access, the address of an element in the data block is assigned dynamically and automatically applied to the HMI tag in case of a change. You do not need to compile the connected data block or the WinCC project for this step.
  For data blocks with optimized access, only symbolic addressing is available.
- Symbolic addressing of data blocks with standard access
  For symbolic addressing of a data block with standard access, the address of an element in the data block is assigned permanently. The valid data type for the HMI tag is automatically selected by the system. Any change in the address of an element in the data block is applied directly to the HMI tag. You do not need to compile the connected data block or the WinCC project for this step.
  For data blocks with standard access, you can use symbolic addressing as well as absolute addressing.

For symbolic addressing of elements in a data block, you only need to recompile and reload the WinCC project in case of the following changes:

- If the name or the data type of the connected data block has changed.
- If the name or the data type of a higher-level structure node of the connected element in the data block has changed.
- If the name of the connected data block has changed.

Symbolic addressing is currently only available on PLCs of the SIMATIC S7 1200 type. Addressing with optimized access is only available in an integrated connection.

You can also use absolute addressing with an integrated connection. You have to use absolute addressing for PLC tags from a SIMATIC S7 300/400 PLC. If you have connected an HMI tag with a PLC tag and the address of the PLC tag changes, you only have to recompile the control program to update the new address in WinCC. Then you recompile the WinCC project and load it onto the HMI device.

In WinCC, symbolic addressing is the default method. To change the default setting, select the menu command "Options > Settings". Select "Visualization > Tags" in the "Settings" dialog. If required, disable the "Symbolic access" option.

The availability of an integrated connection depends on the PLC used. The following table shows the availability:

PLC	Integrated connection	Comments
S7 300/400	Yes	The linking of tags is not checked in Runtime. If the tag address changes in the PLC and the HMI device is not compiled again and loaded, the change is not registered in runtime.
S7 1200	Yes	A validity check of the tag connection is performed in runtime during symbolic addressing. If an address is changed in the PLC, the change is registered and an error message is issued. The reaction described for S7 300/400 applies whilst addressing with standard access.

Create an integrated connection in the "Devices & Networks" editor. If the PLC is contained in the project and integrated connections are supported, you can then also have the connection created automatically. To do this, when configuring the HMI tag, simply select an existing PLC tag to which you want to connect the HMI tag. The integrated connection is then automatically created by the system.

Addressing with non-integrated connections

In the case of a project with a non-integrated connection, you always configure a tag connection with absolute addressing. Select the valid data type yourself. If the address of a PLC tag changes in a project with a non-integrated connection during the course of the project, you also have to make the change in WinCC. The tag connection cannot be checked for validity in Runtime, an error message is not issued.

A non-integrated connection is available for all supported PLCs.

Symbolic addressing is not available in a non-integrated connection.

With a non-integrated connection, the control program does not need to be part of the WinCC project. You can perform the configuration of the PLC and the WinCC project independently of each other. For configuration in WinCC, only the addresses used in the PLC and their function have to be known.

See also

Basics of communication (Page 2381)

Internal Tags

Introduction

Internal tags do not have any connection to the PLC.

Principle

Internal tags are stored in the memory of the HMI device. Therefore, only this HMI device has read and write access to the internal tags. You can create internal tags to perform local calculations, for example.

You can use the HMI data types for internal tags. Availability depends on the HMI device being used.

The following HMI data types are available:

HMI data type	Data format
Array	One-dimensional array
Bool	Binary tag
DateTime	Date/time format
DInt	Signed 32-bit value
Int	Signed 16-bit value
LReal	Floating-point number 64-bit IEEE 754
Real	Floating-point number 32-bit IEEE 754
SInt	Signed 8-bit value
UDInt	Unsigned 32-bit value
UInt	Unsigned 16-bit value
USInt	Unsigned 8-bit value
WString	Text tag, 16-bit character set

10.8.3.2 Data exchange using area pointers

Basic information on area pointers

Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations. The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

For example, area pointers are required for the following data:

- Recipes
- Job mailboxes
- Sign-of-life monitoring

Area pointers

The following area pointers are supported:

Area pointer

Area pointers can be configured for connections.

- Data record
- Date/time
- Coordination
- Job mailbox

Global area pointers of the HMI device

Global area pointers can be configured for separate connections.

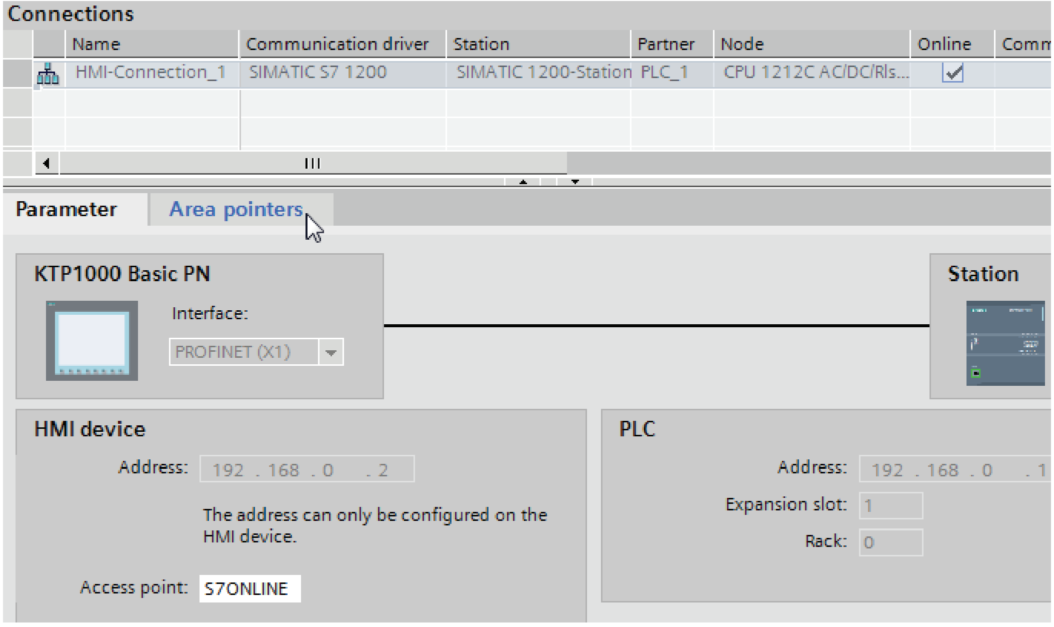
- Screen number
- Date/time PLC
- Project ID

Area pointers for connections

Introduction

Using the "Area pointer" tab of the "Connections" editor, you can configure the usage of the available area pointers.

To configure the area pointers, open the "Connections" editor and open the "Area pointer" tab.



Connections

Name	Communication driver	Station	Partner	Node	Online	Comm
HMI-Connection_1	SIMATIC S7 1200	SIMATIC 1200-Station	PLC_1	CPU 1212C AC/DC/Rls...	☑	

Interface: PROFINET (X1)

HMI device

Address: 192 . 168 . 0 . 2

The address can only be configured on the HMI device.

Access point: S7ONLINE

PLC

Address: 192 . 168 . 0 . 1

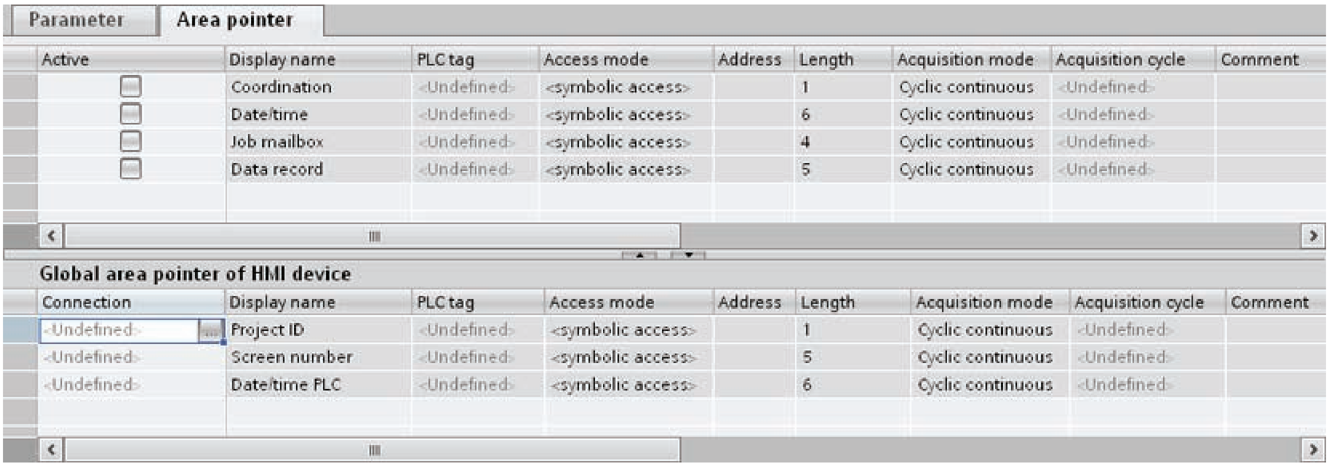
Expansion slot: 1

Rack: 0

Structure

The "Area pointer" tab contains two tables of area pointers. The top part of the table contains the area pointers you can create and enable separately for each available connection.

The "Global area pointers of HMI device" table contains the area pointers which are created only once in the project and can be used for only one connection.



Active	Display name	PLC tag	Access mode	Address	Length	Acquisition mode	Acquisition cycle	Comment
□	Coordination	<Undefined>	<symbolic access>		1	Cyclic continuous	<Undefined>	
□	Date/time	<Undefined>	<symbolic access>		6	Cyclic continuous	<Undefined>	
□	Job mailbox	<Undefined>	<symbolic access>		4	Cyclic continuous	<Undefined>	
□	Data record	<Undefined>	<symbolic access>		5	Cyclic continuous	<Undefined>	

Global area pointer of HMI device

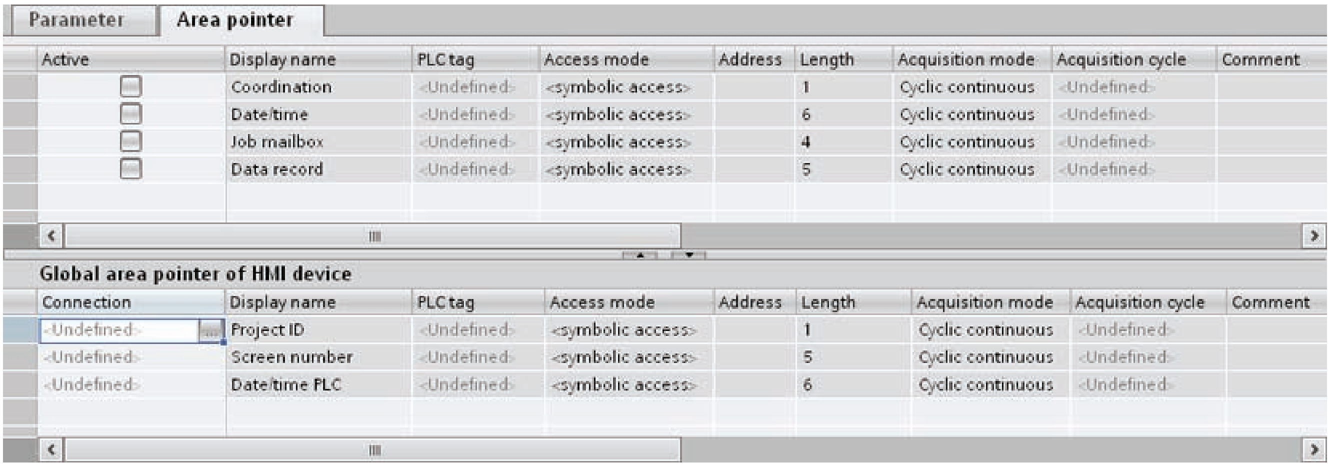
Connection	Display name	PLC tag	Access mode	Address	Length	Acquisition mode	Acquisition cycle	Comment
<Undefined>	Project ID	<Undefined>	<symbolic access>		1	Cyclic continuous	<Undefined>	
<Undefined>	Screen number	<Undefined>	<symbolic access>		5	Cyclic continuous	<Undefined>	
<Undefined>	Date/time PLC	<Undefined>	<symbolic access>		6	Cyclic continuous	<Undefined>	

Use of area pointers

"Area pointer" tab

Before you use an area pointer, you enable it under "Connections > Area pointer". You then assign the area pointer parameters.

You assign the following parameters in the "Area pointer" tab:



Active	Display name	PLC tag	Access mode	Address	Length	Acquisition mode	Acquisition cycle	Comment
□	Coordination	<Undefined>	<symbolic access>		1	Cyclic continuous	<Undefined>	
□	Date/time	<Undefined>	<symbolic access>		6	Cyclic continuous	<Undefined>	
□	Job mailbox	<Undefined>	<symbolic access>		4	Cyclic continuous	<Undefined>	
□	Data record	<Undefined>	<symbolic access>		5	Cyclic continuous	<Undefined>	

Global area pointer of HMI device

Connection	Display name	PLC tag	Access mode	Address	Length	Acquisition mode	Acquisition cycle	Comment
<Undefined>	Project ID	<Undefined>	<symbolic access>		1	Cyclic continuous	<Undefined>	
<Undefined>	Screen number	<Undefined>	<symbolic access>		5	Cyclic continuous	<Undefined>	
<Undefined>	Date/time PLC	<Undefined>	<symbolic access>		6	Cyclic continuous	<Undefined>	

- Active

  Enables the area pointer.

- Pointer name

  Name of the area pointer specified by WinCC.

- PLC tag

  Here you select the PLC tag or the tag array that you have configured as the data area for the area pointer.
- Access mode

  Here you can select from the following access modes:
  - Symbolic access
  - Absolute access
- Address

  If you selected "Symbolic access", no address is entered in this field.

  If you selected "Absolute access", enter the address of a tag in the "Address" field.
- Length

  WinCC specifies the length of the area pointer.
- Acquisition cycle

  You specify the acquisition cycle in this field for area pointers that are read by the HMI device. Note that a very short acquisition time may have a negative impact on HMI device performance.
- Comment

  Enter a comment, for example, to describe the purpose of the area pointer.

Accessing data areas

Accessing data areas

The following table shows how HMI devices and PLCs access individual data areas for read (R) or write (W) operations.

Data area	Required for	HMI device	PLC
Screen number	Evaluation by the PLC in order to determine the active screen.	W	R
Data record	Transfer of data records with synchronization	R/W	R/W
Date/time	Transfer of the date and time from the HMI device to the PLC	W	R
Date/time PLC	Transfer of the date and time from the PLC to the HMI device	R	W
Coordination	Requesting the HMI device status in the PLC program	W	R
Project ID	Runtime checks for consistency between the WinCC project ID and the project in the PLC	R	W
Job mailbox	Triggering of HMI device functions by the PLC program	R/W	R/W

Configuring area pointers

Configuration of area pointers

Introduction

You use an area pointer to access a data area in the PLC. The data area is stored in the PLC.

Prior to configuring area pointers

Before you use the area pointer, you must enable and parameterize it under "Connections > Area Pointer".

Global data block

To access the data area in the PLC, you have to create a global data block in the PLC program. The following example shows the use of a data block.

Length of area pointers

For area pointers with a length >= 1, you set up the data area as a tag array in a global data block or instance data block.

You also have the option to use a PLC tag for area pointers with a length = 1.

The configuration of the tags in a data block is dependent on the length of the area pointer you want to use. The unit for the length of an area pointer is a 16-bit word.

If, for example, you want to use an area pointer with a length of "5", you must create an array with 5 array elements in the data block.

Alternative procedure

Alternatively, you can also use the absolute access mode to access area pointers.

Parameterizing a global data block

Introduction

To access the data area in the PLC, a global data block for the area pointer must be parameterized in the PLC program.

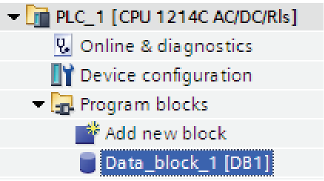
Requirements

- A PLC is created in the project.
- A connection is configured between the PLC and the HMI device.
- The PLC program contains a global data block.

Procedure

1. Open "PLC > Program blocks" in the project tree.
2. Double-click the global data block you created previously.

   The data block opens.



PLC_1 [CPU 1214C AC/DC/Rls]
Online & diagnostics
Device configuration
Program blocks
Add new block
Data_block_1 [DB1]

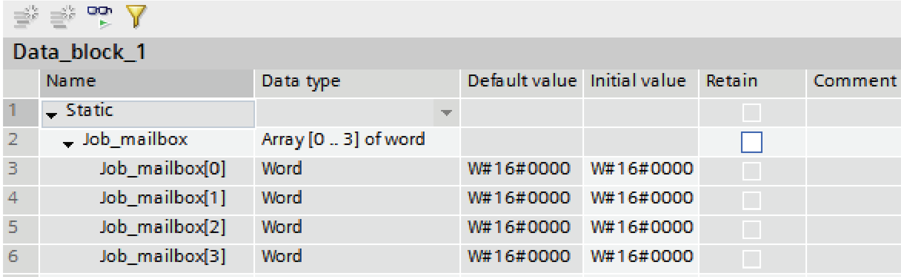
3. Enter a tag name in the "Name" column.
4. Select "Array[lo .. hi] of type" as the data type in the "Data type" column.
5. Replace the "lo" entry by the low value for the dimension of the array.
6. Replace the "hi" entry by the high value for the dimension of the array.

   Example: If you configure an area pointer with the length "4", enter the value "0" for "lo" and the value "3" for "hi" inside the brackets.
7. Replace the "type" designation with the "word" data type.

   The full data type for an array of 4 tags appears as follows: "Array[0 .. 3] of word".

   The tag array is created after the entry is confirmed.
8. Click "Compile".

   The project is compiled.



Data_block_1

	Name	Data type	Default value	Initial value	Retain	Comment
1	Static				□	
2	Job_mailbox	Array [0 .. 3] of word			□	
3	Job_mailbox[0]	Word	W#16#0000	W#16#0000	□	
4	Job_mailbox[1]	Word	W#16#0000	W#16#0000	□	
5	Job_mailbox[2]	Word	W#16#0000	W#16#0000	□	
6	Job_mailbox[3]	Word	W#16#0000	W#16#0000	□	

Configuring an area pointer for a connection

Introduction

After you have parameterized the global data block, you now create the area pointer for the connection.

Requirements

- The global data block has been parameterized in the PLC program.

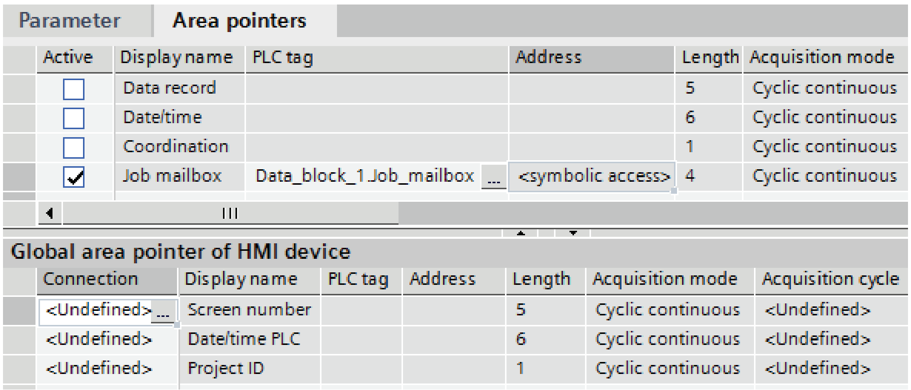
Procedure

1. Open "HMI >Connections" in the project tree.
2. Click the "Area pointer" tab.
3. Enable the required area pointer.

   You enable a global area pointer by selecting the connection in the "Connection" field.
4. Click the navigation button in the "PLC tag" field.

   The object list opens.
5. Navigate to the data block in the object list, and select the tag in the right window.

   You do not need an array tag to configure an area pointer with the length of "1".



Active	Display name	PLC tag	Address	Length	Acquisition mode
□	Data record			5	Cyclic continuous
□	Date/time			6	Cyclic continuous
□	Coordination			1	Cyclic continuous
☑	Job mailbox	Data_block_1.Job_mailbox	<symbolic access>	4	Cyclic continuous

Global area pointer of HMI device

Connection	Display name	PLC tag	Address	Length	Acquisition mode	Acquisition cycle
<Undefined>	Screen number			5	Cyclic continuous	<Undefined>
<Undefined>	Date/time PLC			6	Cyclic continuous	<Undefined>
<Undefined>	Project ID			1	Cyclic continuous	<Undefined>

6. Select the "Word" data type when creating the tag in the data block.

If required, set additional parameters, such as the acquisition cycle, during configuration.

Result

The area pointer is enabled and connected to the PLC tag in the global data block.

10.8.4 Device dependency

10.8.4.1 Basic Panel

Communication drivers for Basic Panels

Device dependency of the Basic Panels

The following table shows which communication drivers you can configure with the various Basic Panels.

Communication drivers

HMI devices	SIMATIC S7 1200	SIMATIC S7 300/400	SIMATIC S7 200	SIMATIC HTTP protocol	OPC	Allen-Bradley EtherNet/IP	Allen-Bradley DF1	Mitsubishi MC TCP/IP	Mitsubishi FX	Modicon Modbus TCP/IP	Modicon Modbus RTU	Omron Host Link
KP300 Basic	Yes	Yes	Yes	No	No	Yes	No	Yes	No	Yes	No	No
KTP400 Basic mono PN	Yes	Yes	Yes	No	No	Yes	No	Yes	No	Yes	No	No
KTP400 Basic mono PN Portrait	Yes	Yes	Yes	No	No	Yes	No	Yes	No	Yes	No	No
KTP600 Basic DP	Yes	Yes	Yes	No	No	No	Yes [2)]	No	Yes	No	yes [1)]	Yes
KTP600 Basic DP Portrait	Yes	Yes	Yes	No	No	No	Yes [2)]	No	Yes	No	yes [1)]	Yes
KTP600 Basic PN	Yes	Yes	Yes	No	No	Yes	No	Yes	No	Yes	No	No
KTP600 Basic PN Portrait	Yes	Yes	Yes	No	No	Yes	No	Yes	No	Yes	No	No
KTP600 Basic mono PN	Yes	Yes	Yes	No	No	Yes	No	Yes	No	Yes	No	No

HMI devices	SIMATIC S7 1200	SIMATIC S7 300/400	SIMATIC S7 200	SIMATIC HTTP protocol	OPC	Allen-Bradley EtherNet/IP	Allen-Bradley DF1	Mitsubishi MC TCP/IP	Mitsubishi FX	Modicon Modbus TCP/IP	Modicon Modbus RTU	Omron Host Link
KTP600 Basic mono PN Portrait	Yes	Yes	Yes	No	No	Yes	No	Yes	No	Yes	No	No
KTP1000 Basic DP	Yes	Yes	Yes	No	No	No	Yes [2]	No	Yes	No	yes [1]	Yes
KTP1000 Basic PN	Yes	Yes	Yes	No	No	Yes	No	Yes	No	Yes	No	No
TP1500 Basic PN	Yes	Yes	Yes	No	No	Yes	No	Yes	No	Yes	No	No

[1] only with RS 422-RS232 converter

Order number: 6AV6 671-8XE00-0AX0

[2] Direct communication with PLC 5 or KF2 module, otherwise only approved with RS422-RS232 converter (option).

Order number: 6AV6 671-8XE00-0AX0

Interfaces of the Basic Panels

Device dependency of the Basic Panels

The following table shows which HMI device interfaces are available for the communication driver protocols.

Table 10- 11 Basic Panels

	KTP300 Basic	KTP400 Basic PN	KTP600 Basic DP	KTP600 Basic PN	KTP1000 Basic DP	KTP1000 Basic PN	TP1500 Basic PN
SIMATIC S7 - PPI [1]	—	—	MPI/DP (X2)	—	MPI/DP (X2)	—	—
SIMATIC S7 - MPI	—	—	MPI/DP (X2)	—	MPI/DP (X2)	—	—
SIMATIC S7 - PROFIBUS	—	—	MPI/DP (X2)	—	MPI/DP (X2)	—	—
SIMATIC S7 - PROFINET	PROFINET (X1)	PROFINET (X1)	—	PROFINET (X1)	—	PROFINET (X1)	PROFINET (X1)
SIMATIC HMI HTTP protocol	—	—	—	—	—	—	—
OPC	—	—	—	—	—	—	—
Allen-Bradley EtherNet/IP	PROFINET (X1)	PROFINET (X1)	—	PROFINET (X1)	—	PROFINET (X1)	PROFINET (X1)
Allen-Bradley DF1	—	—	MPI/DP (X2) [2]	—	MPI/DP (X2) [2]	—	—
Mitsubishi TCP/IP	PROFINET (X1)	PROFINET (X1)	—	PROFINET (X1)	—	PROFINET (X1)	PROFINET (X1)
Mitsubishi FX	—	—	MPI/DP (X2) (RS422)	—	MPI/DP (X2) (RS422)	—	—
Modicon Modbus TCP	PROFINET (X1)	PROFINET (X1)	—	PROFINET (X1)	—	PROFINET (X1)	PROFINET (X1)
Modicon Modbus RTU	—	—	MPI/DP (X2) [3]	—	MPI/DP (X2) [3]	—	—
Omron Host Link	—	—	MPI/DP (X2) (RS422)	—	MPI/DP (X2) (RS422)	—	—

[1] For SIMATIC S7-200 only

[2] Direct communication with PLC5 or KF2 module, otherwise only approved with RS422-RS232 converter (option).

Order number: 6AV6 671-8XE00-0AX0

[3] only approved with RS 422-RS232 converter

Order number: 6AV6 671-8XE00-0AX0

Area pointers for Basic Panels

Introduction

Area pointers are parameter fields from which the HMI device obtains information about the location and size of data areas in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations. Based on the evaluation of data stored in the data areas, the PLC and HMI device initiate mutually defined actions.

WinCC uses the following area pointers:

- Job mailbox
- Project ID
- Screen number
- Data record
- Date/time
- Date/time PLC
- Coordination

Availability of the area pointers

The following table shows the availability of the area pointers on the HMI devices. Note that the area pointers can be used only for available communication drivers.

Area pointer

	KP300 Basic	KTP400 Basic PN	KTP600 Basic PN	KTP600 Basic DP	KTP1000 Basic PN	KTP1000 Basic DP	TP1500 Basic PN
Screen number	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Data record	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Date/time	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Date/time PLC	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Coordination	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Project ID	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Job mailbox	Yes	Yes	Yes	Yes	Yes	Yes	Yes

10.8.5 Communicating with SIMATIC S7 1200

10.8.5.1 Communication with SIMATIC S7 1200

Introduction

This section describes the communication between an HMI device and the SIMATIC S7 1200 PLC.

You can configure the following communication channels for the SIMATIC S7 1200 PLC:

- PROFINET
- PROFIBUS

HMI connection for communication

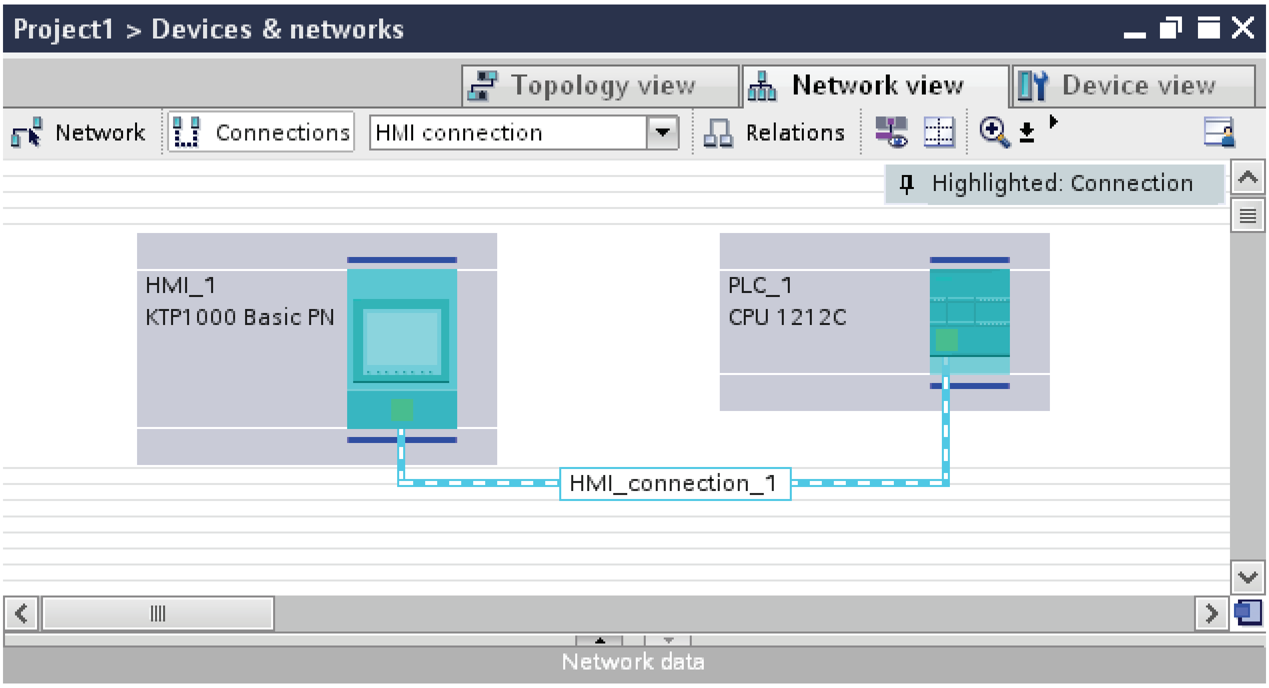
You configure connections between the HMI device and a SIMATIC S7 1200 in the "Devices & Networks" editor. If you have configured a HMI device with a serial port, you must configure a PROFIBUS-capable communication module to the SIMATIC S7 1200.

10.8.5.2 Communication via PROFINET

Communication via PROFINET

HMI connections via PROFINET

If you have inserted an HMI device and a SIMATIC S7 1200 into the project, you interconnect the two PROFINET interfaces in the "Devices & Networks" editor.



You can also connect multiple HMI devices to one SIMATIC S7 1200 and multiple SIMATIC S7 1200s to one HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFINET in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFINET in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFINET

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 1200 via PROFINET or Ethernet in the "Devices & Networks" editor.

> ⚠ CAUTION
>
> **Communication via Ethernet**
>
> In Ethernet-based communication, the end user is responsible for the security of his data network.
>
> Targeted attacks can overload the device and interfere with proper functioning.

Requirements

The following communication partners are created in the "Devices & Networks" editor:

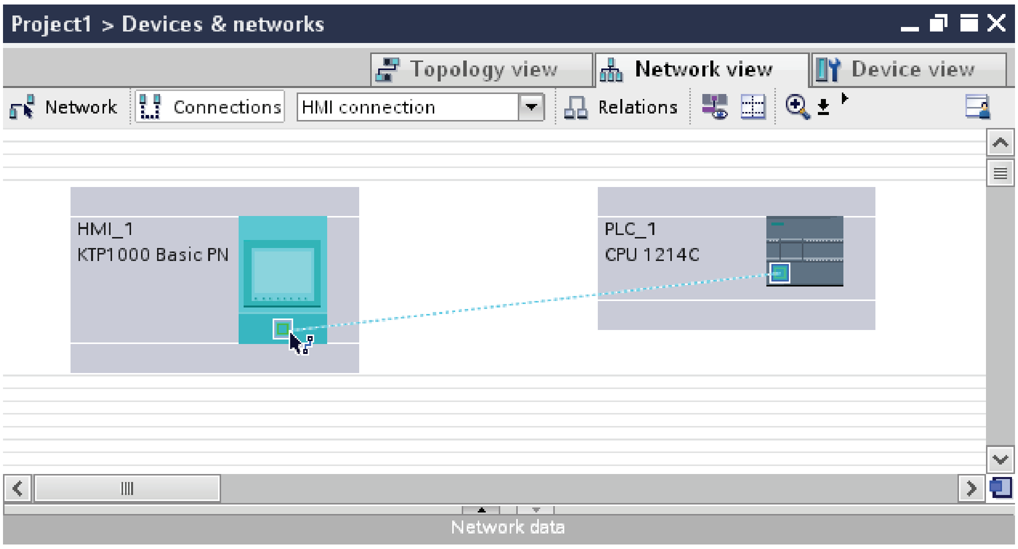
- SIMATIC S7 1200
- HMI device with PROFINET or Ethernet interface

Procedure

1. Double-click the "Devices & Networks" item in the project tree.

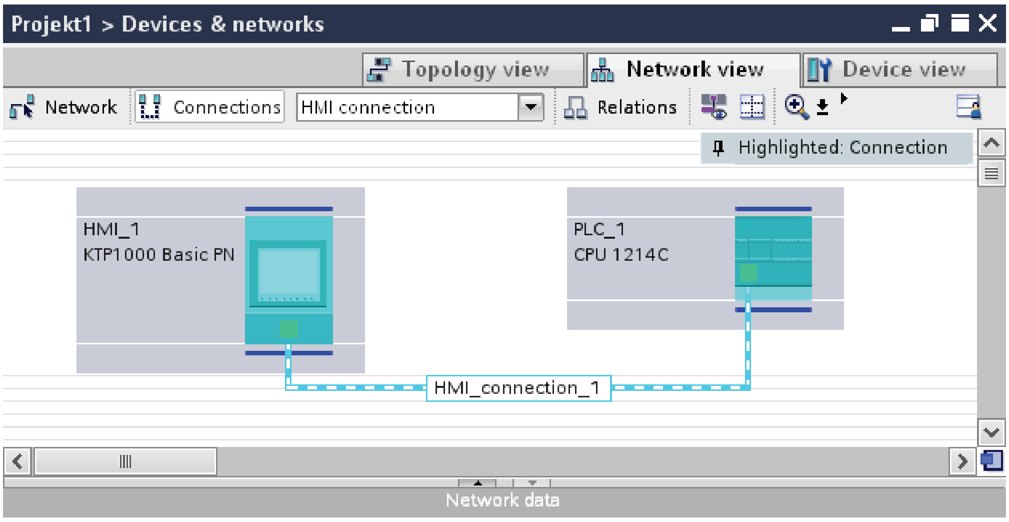
   The available communication partners in the project are displayed graphically in the network view.
2. Click the "Connections" button and select "HMI connection" for the connection type.

   The devices available for connection are highlighted in color.
3. Click the PROFINET interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFINET or Ethernet interface of the HMI device.



4. Click the connecting line.

5. Click "Highlight HMI connection" and select the HMI connection.



The connection is displayed graphically in the Inspector window.

6. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.

   See the chapter "PROFINET parameters (Page 2421)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC S7 1200. The IP address and subnet mask connection parameters are configured.

PROFINET parameters

PROFINET parameters for the HMI connection

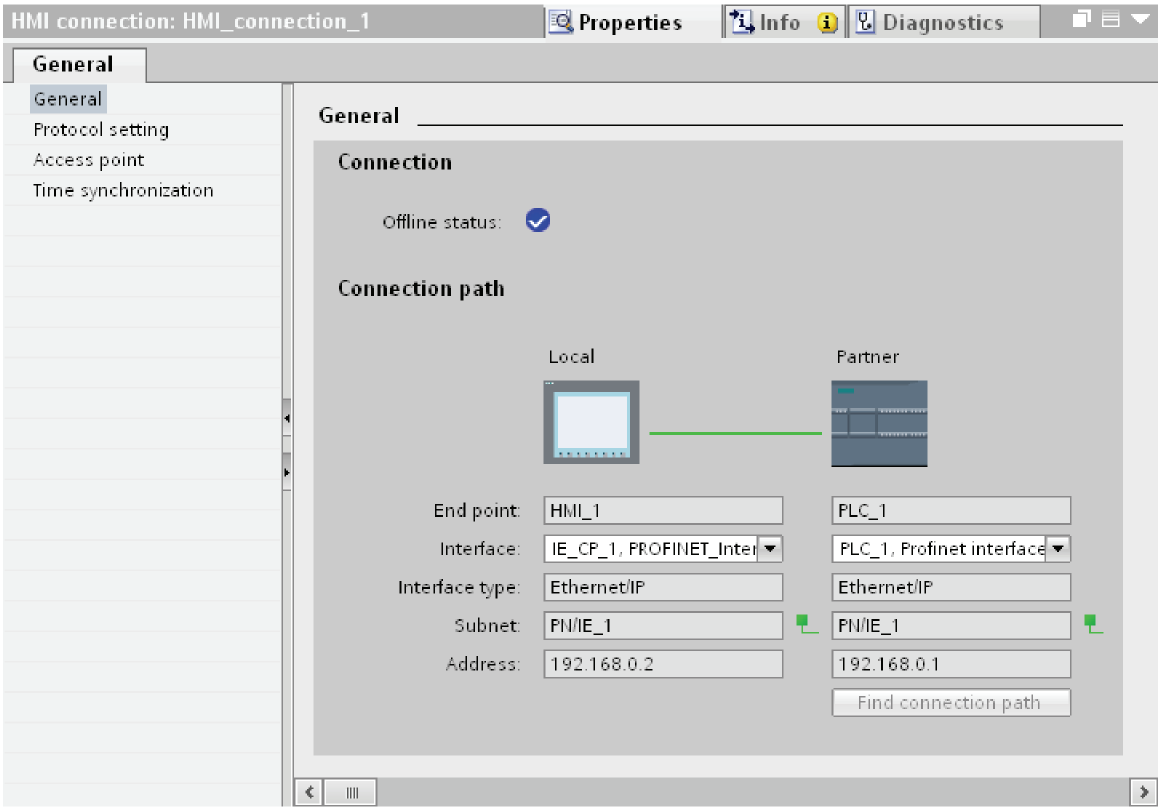
PROFINET parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and changing the HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



Offline status: ✓

	Local	Partner
End point:	HMI_1	PLC_1
Interface:	IE_CP_1, PROFINET_Inter	PLC_1, Profinet interface
Interface type:	Ethernet/IP	Ethernet/IP
Subnet:	PN/IE_1	PN/IE_1
Address:	192.168.0.2	192.168.0.1

"Connection"

Displays whether the devices are networked together.

✓ - displayed if the devices are networked together.

❗ - displayed if the devices are not networked together.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFINET parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"

  Displays the device name. This area is not editable.

- "Interface"

  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.

- "Interface type"

  Displays the selected interface type. This area cannot be edited.

- "Subnet"

  Displays the selected subnet. This area cannot be edited.

- "Address"

  Displays the selected IP address of the device. This area cannot be edited.

- "Find connection path" button

  Enables the subsequent specification of connections.

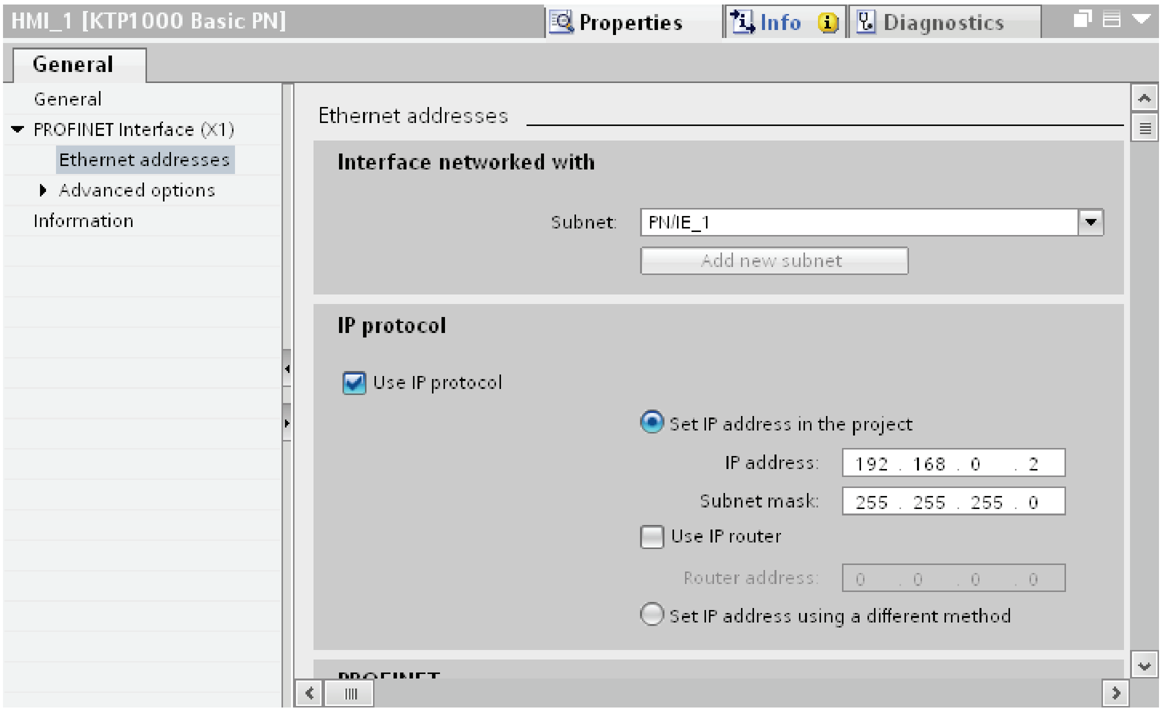
PROFINET parameters for the HMI device

PROFINET parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFINET parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General > General".



Interface networked with

Subnet: PN/IE_1

IP protocol

☑ Use IP protocol

● Set IP address in the project

IP address: 192 . 168 . 0 . 2

Subnet mask: 255 . 255 . 255 . 0

□ Use IP router

Router address: 0 . 0 . 0 . 0

○ Set IP address using a different method

"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Set IP address in the project"

  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.

  Note

  The device is automatically restarted in the case of HMI devices with the Windows CE 3.0 operating system.

  HMI devices with Windows CE 3.0:

  - OP 77B
  - TP 177B color PN/DP
  - TP 177B mono DP
  - OP 177B color PN/DP
  - OP 177B mono DP
  - Mobile Panel 177 PN
  - Mobile Panel 177 DP
  - TP 277 6"
  - OP 277 6"

- "Subnet mask"

  You assign data of the subnet mask in the "Subnet mask" area.

- "Use IP router"

  If you are using an IP router, select "Use IP router" and enter the router address in the "Router address" field.

- "Set IP address using a different method"

  If the function "Set IP address using a different method" is activated, the IP address is not taken from the project. You have to enter the IP address directly in the Control Panel of the HMI device.

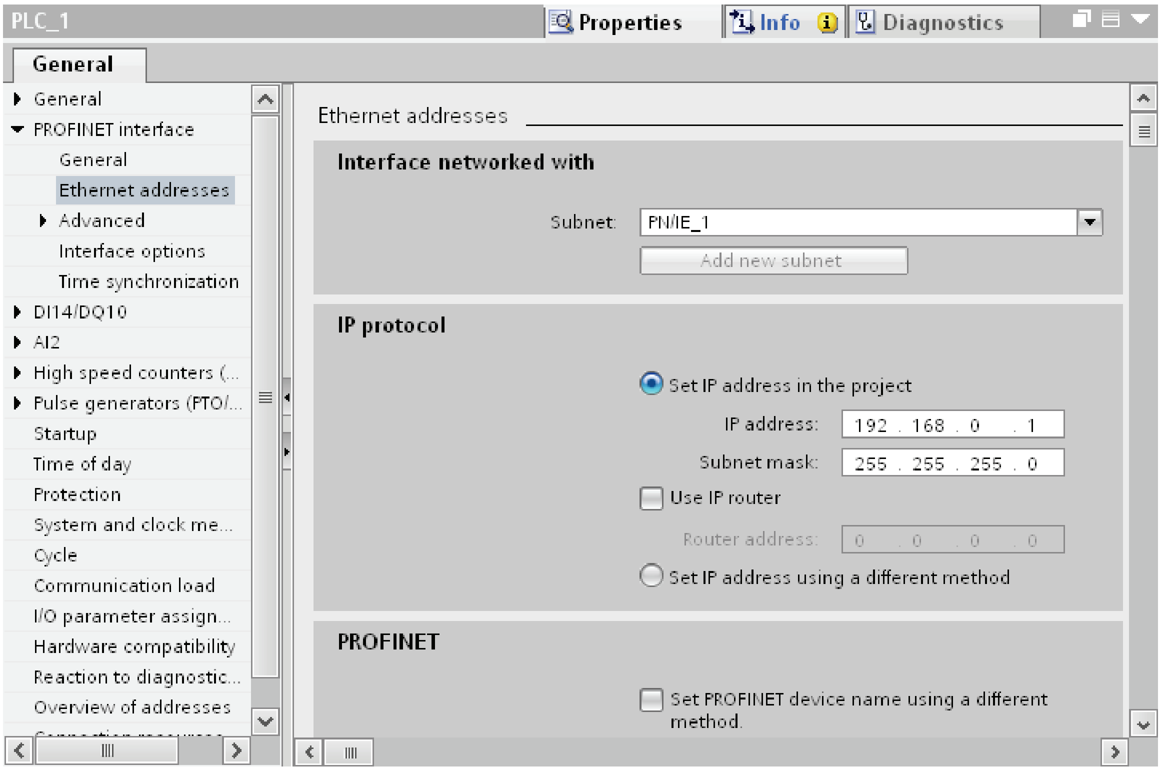
PROFINET parameters for the PLC

PROFINET parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFINET parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



Ethernet addresses

Interface networked with

Subnet: PN/IE_1

IP protocol

● Set IP address in the project

IP address: 192 . 168 . 0 . 1

Subnet mask: 255 . 255 . 255 . 0

□ Use IP router

Router address: 0 . 0 . 0 . 0

○ Set IP address using a different method

PROFINET

□ Set PROFINET device name using a different method.

"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Interface type"

  Depending on the HMI device type, you have various interfaces to choose from.
- "IP address"

  You assign the IP address of the HMI device in the "IP address" area.
- "Subnet mask"

  You assign data of the subnet mask in the "Subnet mask" area.

  If you are using an IP router, select "Use IP router" and enter the router address in the field.

Configuring Industrial Ethernet

Rules for the network configuration

The Ethernet interfaces of the devices have a default IP address that you can change.

IP address

The IP parameters are visible if the communication-capable devices support the TCP/IP protocol.

The IP address consists of 4 decimal figures in the range of 0 to 255. The decimal figures are separated from one another by a dot.

Example: 140.80.0.2

The IP address consists of the following:

- The address of the (sub) net
- The address of the node (generally also called host or network node)

Subnet mask

The subnet mask splits these two addresses. It determines which part of the IP address addresses the network and which part of the IP address addresses the node.

The set bits of the subnet mask determine the network part of the IP address.

Example:

Subnet mask: 255.255.0.0 = 11111111.11111111.00000000.00000000

In the example given for the above IP address, the subnet mask shown here has the following meaning:

The first 2 bytes of the IP address identify the subnet - i.e. 140.80. The last two bytes address the node, thus 0.2.

It is generally true that:

- The network address results from AND linking the IP address and subnet mask.
- The node address results from AND NOT linking the IP address and subnet mask.

Relation between IP address and default subnet mask

An agreement exists relating to the assignment of IP address ranges and so-called "Default subnet masks". The first decimal number (from the left) in the IP address determines the structure of the default subnet mask. It determines the number of "1" values (binary) as follows:

IP address (decimal)	IP address (binary)	Address class	Default subnet mask
0 to 126	0xxxxxxx.xxxxxxxx....	A	255.0.0.0
128 to 191	10xxxxxx.xxxxxxxx...	B	255.255.0.0
192 to 223	110xxxxx.xxxxxxxx...	C	255.255.255.0

Note

Range of values for the first decimal point

A value of between 224 and 255 is also possible for the first decimal number of the IP address (address class D etc). This is, however, not recommended because there is no address check for these values.

Masking other subnets

You can use the subnet mask to add further structures and form "private" subnets for a subnet that is assigned one of the address classes A, B or C. This is done by setting other lower points of the subnet mask to "1". For each bit set to "1", the number of "private" networks doubles and the number of nodes they contain is halved. Externally, the network functions like an individual network as it did previously.

Example:

You have a subnet of address class B (e.g. IP address 129.80.xxx.xxx) and change the default subnet mask as follows:

Masks	Decimal	Binary
Default subnet mask	255.255.0.0	11111111.11111111.00000000.00000000
Subnet mask	255.255.128.0	11111111.11111111.10000000.00000000

Result:

All nodes with addresses between 129.80.001.xxx and 129.80.127.xxx are on one subnet, all nodes with addresses between 129.80.128.xxx and 129.80.255.xxx are on another subnet.

Router

The job of the routers is to connect the subnets. If an IP datagram is to be sent to another network, it first has to be conveyed to a router. To make this possible, in this case you have to enter the address of the router for each node in the subnet.

The IP address of a node in the subnet and the address of the router may only differ at the points at which there is a "0" in the subnet mask.

Setting port options

Setting the port options

Changing connection settings for the PROFINET IO port

You can change the network settings for the PROFINET IO port as required. By default, the settings are made automatically. In normal situations, this guarantees problem-free communication.

Possible settings for transmission rate / duplex

Depending on the selected device, you can make the following settings for "Transmission rate / duplex":

- Automatic setting

  Recommended default setting of the port. The transmission settings are automatically "negotiated" with the peer port. The "Enable autonegotiation" option is also enabled as a default, in other words, you can use cross cables or patch cables for the connection.

- TP/ITP at x Mbps full duplex (half duplex)

  Setting of the transmission rate and the full duplex/half duplex mode. The effectiveness depends on the "Enable autonegotiation" setting:

  – Autonegotiation enabled

    You can use both cross cable and patch cable.

  – Autonegotiation disabled

    Make sure that you use the correct cable (patch cable or cross cable)! The port is also monitored with this setting.

- Deactivated

  Depending on the module type, the drop down list box can contain the "- Disabled -" option. This option, for example, allows you to prevent access to an unused port for security reasons. With this setting, diagnostic events are not generated.

"Monitor" option

This option enables or disables port diagnostics. Examples of port diagnostics: The link status is monitored, in other words, the diagnostics are generated during link-down and the system reserve is monitored in the case of fiber optic ports.

Option "Enable autonegotiation "

The autonegotiation setting can only be changed if a concrete medium (for example, TP 100 Mbps full duplex) is selected. Whether or not a concrete medium can be set depends on the properties of the module.

If autonegotiation is disabled, this causes the port to be permanently specified, as for example, is necessary for a prioritized startup of the IO device.

You must make sure the partner port has the same settings because with this option the operating parameters of the connected network are not detected and the data transmission rate and transmission mode can accordingly not be optimally set.

Note

When a local port is connected, STEP 7 makes the setting for the partner port if the partner port supports the setting. If the partner port does not accept the setting, an error message is generated.

Wiring rules for disabled autonegotiation

Requirements

You have made the following settings for the port in question, for example, to accelerate the startup time of the IO device:

- Fixed transmission rate
- Autonegotiation incl. autocrossing disabled

The time for negotiating the transmission rate during startup has been saved.

If you have disabled autonegotiation, you must observe the wiring rules.

Wiring rules for disabled autonegotiation

PROFINET devices have the following two types of ports:

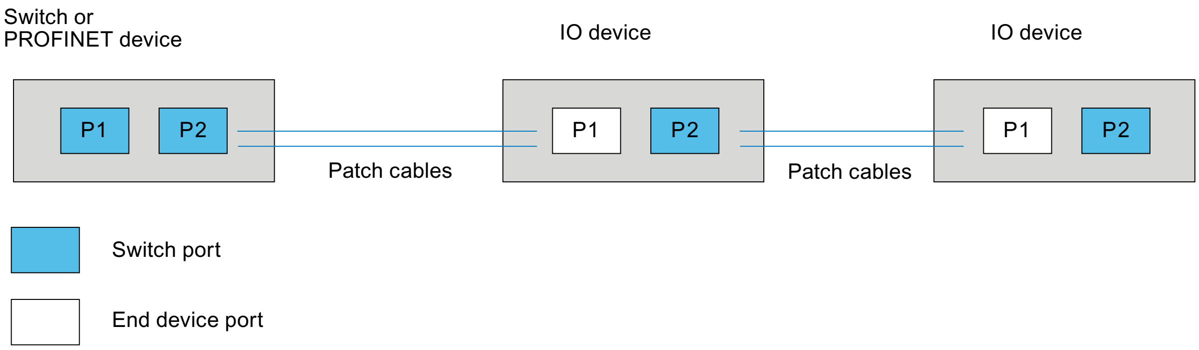
Type of port	PROFINET devices	Note
Switch port with crossed pin assignment	For IO devices: Port 2 For S7 CPUs with 2 ports: Ports 1 and 2	Crossed pin assignment means that the pin assignment for the ports for sending and receiving between the respective PROFINET devices is exchanged internally.
End device port with uncrossed pin assignment	For IO devices: Port 1 For S7 CPUs with one port: Port 1	-

Validity of the wiring rules

The cabling rules described in the following paragraph apply exclusively for the situation in which you have specified a fixed port setting.

Rules for cabling

You can connect several IO devices in line using a single cable type (patch cable). To do this, you connect port 2 of the IO device (distributed I/O) with port 1 of the next IO device. The following graphic gives an example with two IO devices.



Boundaries at the port

Requirements

To use boundaries, the respective device must have more than one port. If the PROFINET does not support boundary settings, they are not shown.

Enable boundaries

"Boundaries" are limits for transmission of certain Ethernet frames. The following boundaries can be set at a port:

- "End of discovery of accessible devices"

  No forwarding of DCP frames to identify accessible devices. Devices downstream from this port cannot be reached by the project tree under "Accessible devices". Devices downstream from this port cannot be reached by the CPU.
- "End of topology discovery"

  LLDP frames (Link Layer Discovery Protocol) are not forwarded for topology detection.
- "End of sync domain"

  No forwarding of sync frames transmitted to synchronize nodes within a sync domain.

  If you operate, for example, a PROFINET device with more than two ports in a ring, you should prevent the sync frame from being fed into the ring by setting a sync boundary (at the ports not inside the ring).

  Additional example: If you want to use several sync domains, configure a sync domain boundary for the port connected to a PROFINET device from the other sync domain.

Restrictions

The following restrictions must be observed:

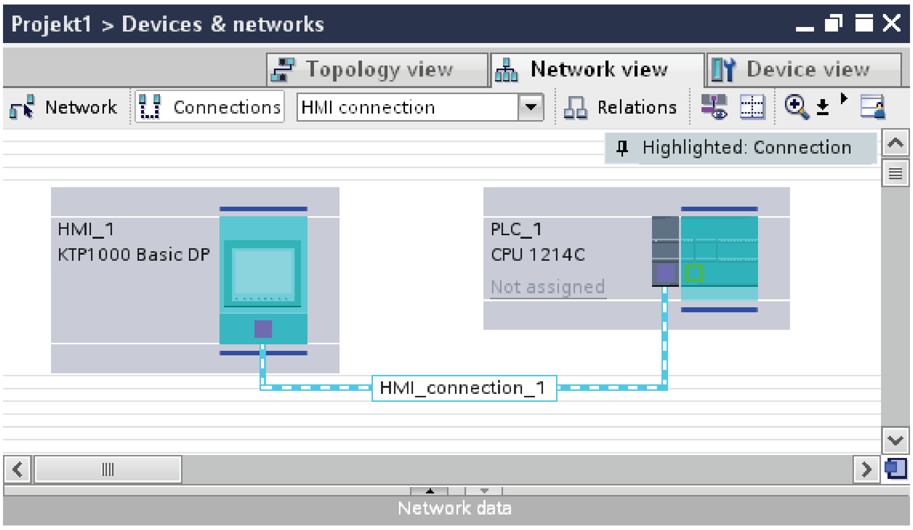
- The individual check boxes can only be used if the port supports the function in question.
- If a partner port has been determined for the port, the following check boxes cannot be used:
  - "End of discovery of accessible devices"
  - "End of topology discovery"
- If autonegotiation is disabled, none of the check boxes can be used.

10.8.5.3 Communication via PROFIBUS

Communication via PROFIBUS

HMI connections via PROFIBUS

If you want to connect a SIMATIC S7 1200 to a HMI device via PROFIBUS, you must configure a PROFIBUS-capable communication module to a slot of the controller first.



HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFIBUS in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFIBUS

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 1200 via PROFIBUS in the "Devices & Networks" editor.

Requirements

The following communication partners are created in the "Devices & Networks" editor:

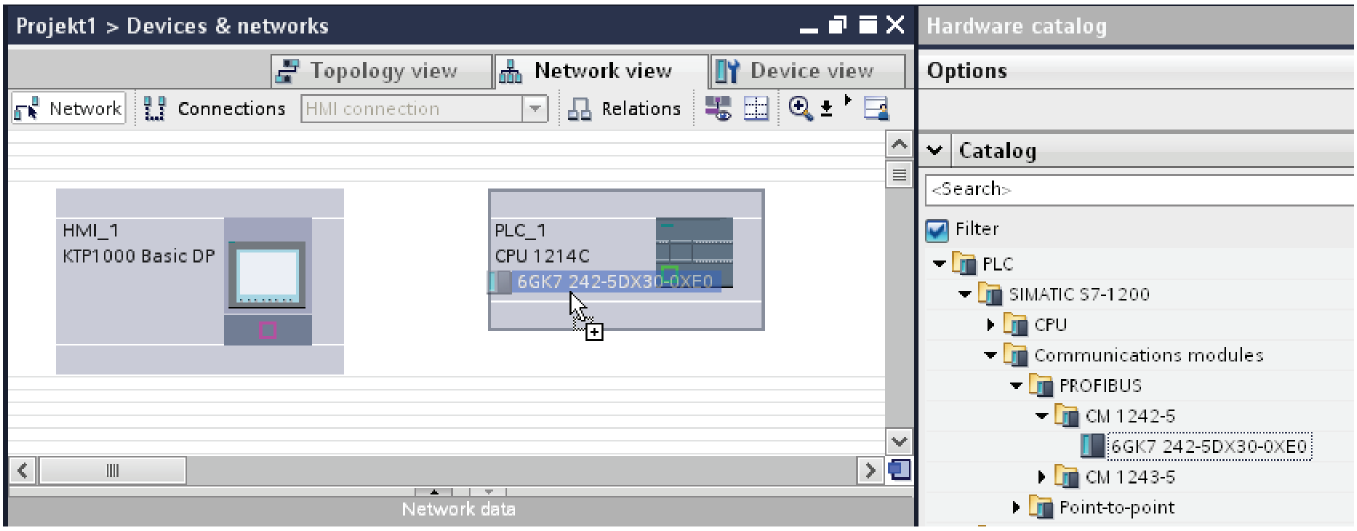
- HMI device with MPI/DP interface
- SIMATIC S7 1200

Procedure

1. Double-click the "Devices & Networks" item in the project tree.

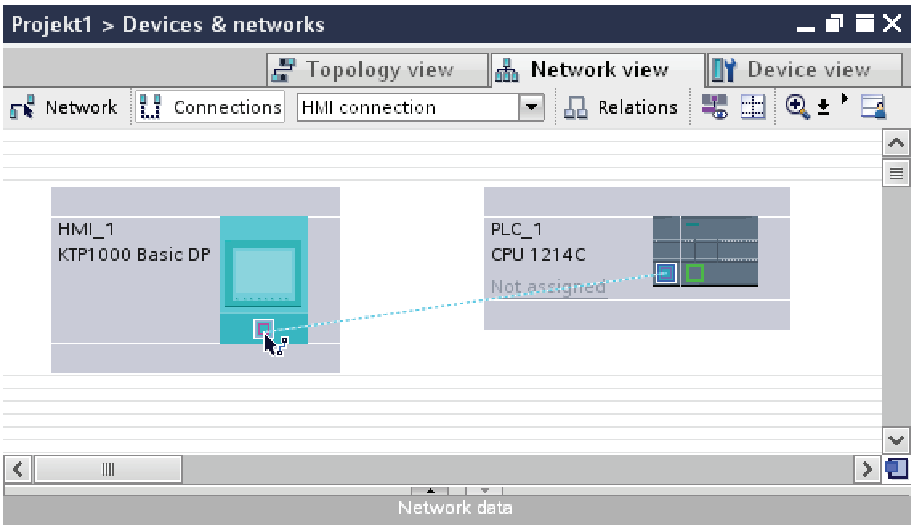
   The available communication partners in the project are displayed graphically in the network view.
2. Click the "Connections" button.

   The devices available for connection are highlighted in color.
3. Use a drag-and-drop operation to move a PROFIBUS-capable communication module from the hardware catalog to the PLC.



4. Click the HMI device interface.
5. Select the "PROFIBUS" interface type in the Inspector window under "Properties > General > PROFIBUS address/ MPI address > Parameters".

6. Click the interface of the communication module and use a drag-and-drop operation to draw a connection to the HMI device.



7. Click the name of the connection.

   The connection is displayed graphically in the Inspector window.
8. Click "Highlight HMI connection" and select the HMI connection.
9. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.

   See the chapter "PROFIBUS parameters (Page 2434)" for additional details.

> **Note**
>
> The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.
>
> You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC S7 1200 via PROFIBUS.

PROFIBUS parameters

PROFIBUS parameters for the HMI connection

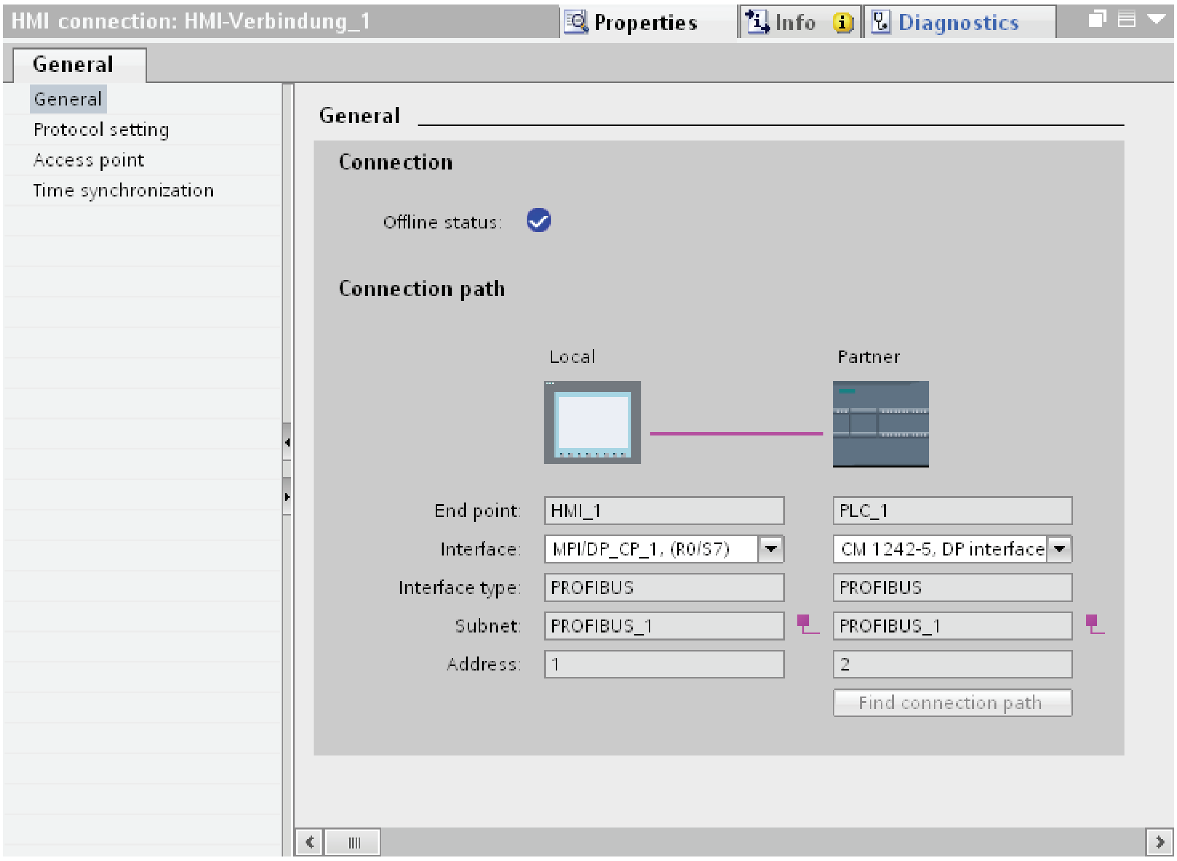
PROFIBUS parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and changing the HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



General

Connection

Offline status: ✓

Connection path

	Local	Partner
End point:	HMI_1	PLC_1
Interface:	MPI/DP_CP_1, (R0/S7)	CM 1242-5, DP interface
Interface type:	PROFIBUS	PROFIBUS
Subnet:	PROFIBUS_1	PROFIBUS_1
Address:	1	2

"Connection"

Displays whether the devices are networked together.

✓ - displayed if the devices are networked together.

❗ - displayed if the devices are not networked together.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFIBUS parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"

  Displays the device name. This area is not editable.
- "Interface"

  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"

Displays the selected interface type. This area is not editable.

- "Subnet"

  Displays the selected subnet. This area is not editable.
- "Address"

  Displays the PROFIBUS address of the device. This area is not editable.
- "Find connection path" button

  Enables the subsequent specification of connections.

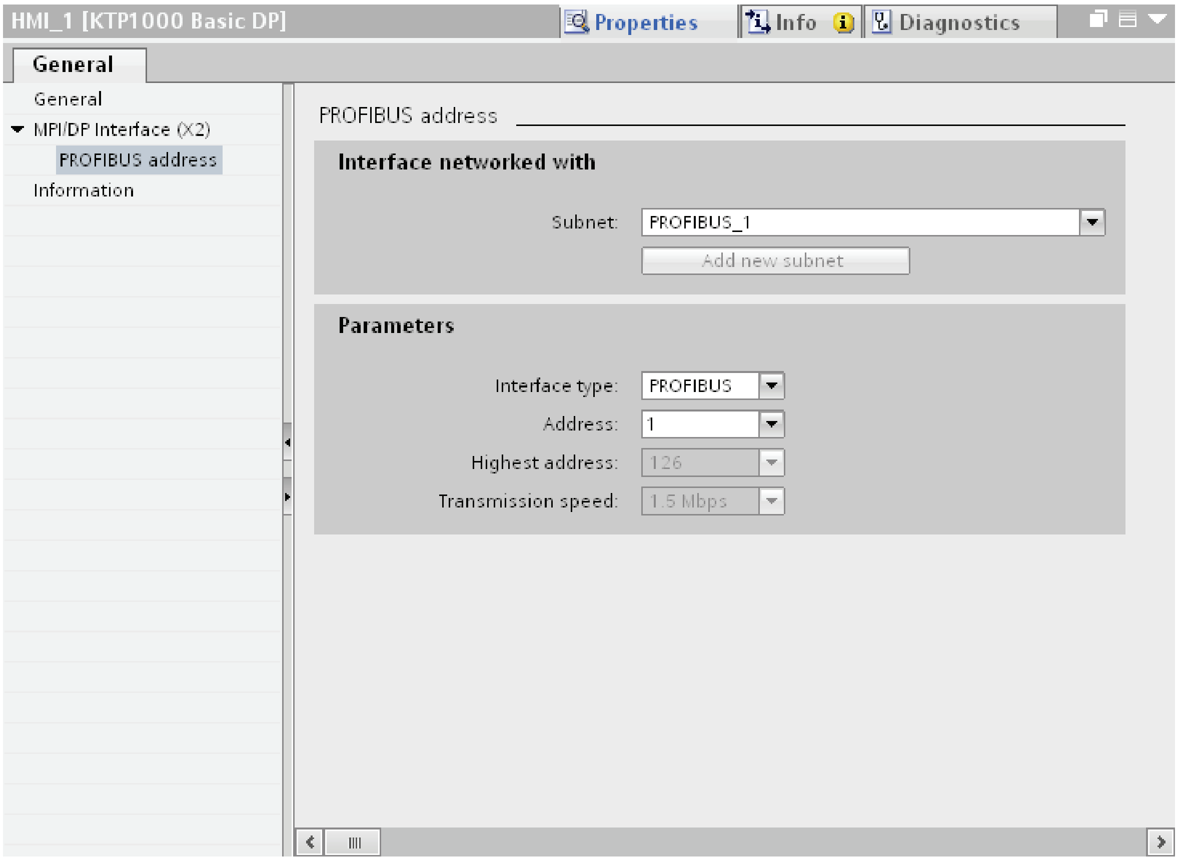
PROFIBUS parameters for the HMI device

PROFIBUS parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFIBUS parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General > General".



Interface networked with

Subnet: PROFIBUS_1

Parameters

Interface type: PROFIBUS

Address: 1

Highest address: 126

Transmission speed: 1.5 Mbps

"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"

  Depending on the HMI device type, you have various interfaces to choose from.
- "Address"

  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.
- "Highest address"

  The "Highest address" area displays the highest address of the PROFIBUS network.
- "Transmission speed"

  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

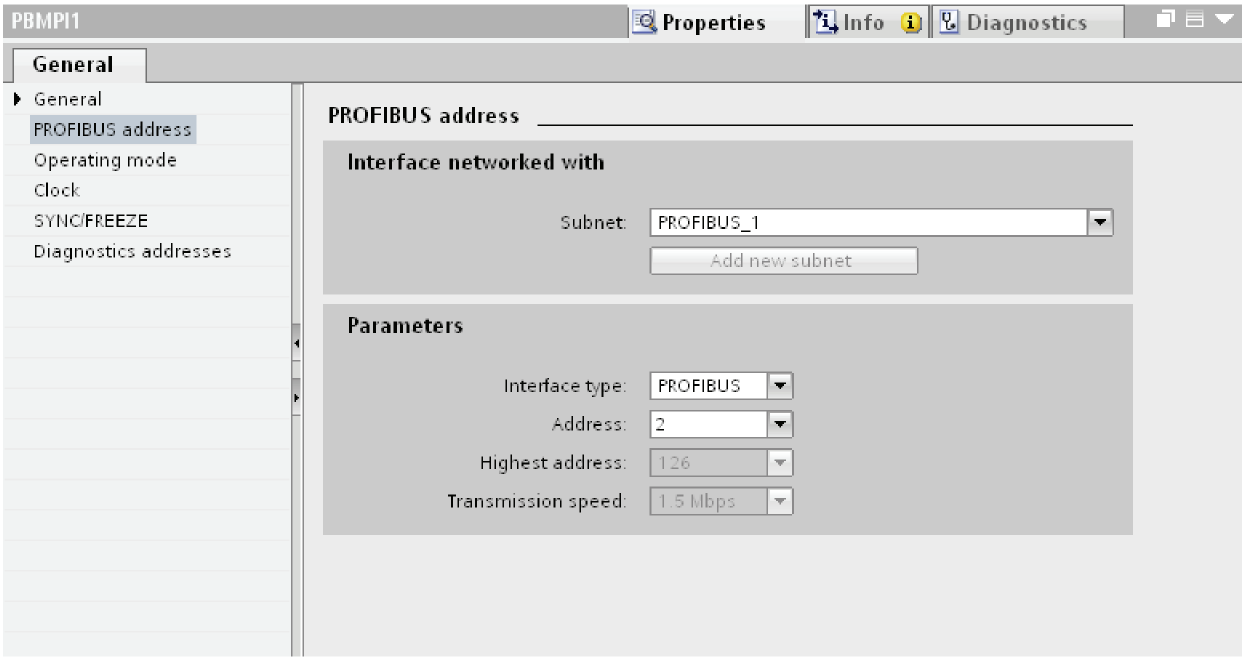
PROFIBUS parameters for the PLC

PROFIBUS parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFIBUS parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



Subnet: PROFIBUS_1

Interface type: PROFIBUS

Address: 2

Highest address: 126

Transmission speed: 1.5 Mbps

"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"

  Depending on the HMI device type, you have various interfaces to choose from.
- "Address"

  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.
- "Highest address"

  The "Highest address" area displays the highest address of the PROFIBUS network.
- "Transmission speed"

  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

Bus profiles with PROFIBUS

Introduction

Depending on the device types connected and protocols used on the PROFIBUS, different profiles are available. The profiles differ in terms of the setting options and calculation of bus parameters. The profiles are explained below.

Devices with different profiles on the same PROFIBUS subnet

The PROFIBUS subnet only functions without problem if the bus parameters of all devices have the same values.

Profiles and transmission rates

Profiles	Supported transmission speeds in Kbits/s
DP	9,6 19,2 45,45 93,75 187,5 500 1500 3000 6000 12000
Standard	9,6 19,2 45,45 93,75 187,5 500 1500 3000 6000 12000
Universal	9,6 19,2 93,75 187,5 500 1500

Meaning of profiles

Profile	Meaning
DP	Select the "DP" bus profile when the only devices connected to the PROFIBUS subnet are those which satisfy the requirements of standard EN 50170 Volume 2/3, Part 8-2 PROFIBUS. The bus parameter setting is optimized on these devices. This includes devices with DP master and DP slave interfaces of the SIMATIC S7 and distributed I/Os of other manufacturers.
Standard	Compared to the "DP" profile, the "Standard" profile also offers scope for devices of another project or devices which have not been configured here to be taken into account when calculating the bus parameters. The bus parameters are then calculated following a simple, non-optimized algorithm.
Universal	Select the "Universal" bus profile when individual devices on the PROFIBUS subnet use the PROFIBUS-FMS service. This includes the following devices for example: • CP 343-5 • PROFIBUS-FMS devices of other manufacturers As with the "Standard" profile, this profile allows you to take other devices into account when calculating the bus parameters.

10.8.5.4 Data exchange

Data exchange using area pointers

General information on area pointers

Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

Configuration of area pointers

Before you use an area pointer, you enable it under "Connections > Area pointer". You then assign the area pointer parameters.

You can find more detailed information on configuring area pointers in:

Configuration of area pointers

Area pointer "Date/time"

Function

This area pointer is used to transfer the date and time from the HMI device to the PLC.

The PLC writes control job "41" to the job mailbox.

When it evaluates the control job, the HMI device writes its current date and the time in the data area configured in the "Date/time" area pointer.

The date/time data area has the following structure:

<table>
<tr><th rowspan="2">Data word</th><th colspan="8">Most significant byte</th><th colspan="8">Least significant byte</th><th rowspan="2"></th></tr>
<tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td><td>0</td><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td><td>0</td></tr>
<tr><td>n+0</td><td colspan="8">Reserved</td><td colspan="8">Hour (0-23)</td><td rowspan="3">Time</td></tr>
<tr><td>n+1</td><td colspan="8">Minute (0-59)</td><td colspan="8">Second (0 to 59)</td></tr>
<tr><td>n+2</td><td colspan="8">Reserved</td><td colspan="8">Reserved</td></tr>
<tr><td>n+3</td><td colspan="8">Reserved</td><td colspan="8">Weekday (1 to 7, 1=Sunday)</td><td rowspan="3">Date</td></tr>
<tr><td>n+4</td><td colspan="8">Day (1 to 31)</td><td colspan="8">Month (1 to 12)</td></tr>
<tr><td>n+5</td><td colspan="8">Year (80 to 99/0 to 29)</td><td colspan="8">Reserved</td></tr>
</table>

Note

Note that when you enter the year, values 80 to 99 result in years 1980 through 1999 and the values 0 to 29 result in the years 2000 through 2029.

Permissible data types

You can use the following data types when you configure the "Date/Time" area pointer.

- Word
- UInt
- DTL

Use of the "DTL" data type

Use data type "DTL" with communication driver S7 1200. A tag of the "DTL" data type has a length of 12 bytes and saves information on date and time in a predefined structure.

The "DTL" data type has the following structure:

Byte	Component	Data type	Value range
0	Year	UINT	1970 to 2554
1			
2	Month	USINT	0 to 12
3	Tag	USINT	1 to 31
4	Day of week	USINT	1(Sunday) to 7(Saturday) The weekday is not considered in the value entry.
5	Hour	USINT	0 to 23
6	Minute	USINT	0 to 59
7	Second	USINT	0 to 59
8	Nanoseconds	UDINT	0 to 999 999 999
9			
10			
11			

"Date/time PLC" area pointer

Function

This area pointer is used to transfer the date and time from the PLC to the HMI device. Use this area pointer if the PLC is the time master.

The PLC loads the data area of the area pointer.

The HMI device reads the data cyclically within the configured acquisition cycle and synchronizes itself.

Note

Set an acquisition cycle of sufficient length for the date/time area pointer PLC to avoid any negative impact on HMI device performance.
Recommended: Acquisition cycle of 1 minute, if the process allows this.

Date/time PLC is a global area pointer and can be configured only once in a project.

The date/time data area has the following structure:

DATE_AND_TIME format (in BCD code)

Data word	Most significant byte			Least significant byte		
	7	. . . . . .	0	7	. . . . . .	0
n+0	Year (80-99/0-29)			Month (1 to 12)		
n+1	Day (1 to 31)			Hour (0 to 23)		
n+2	Minute (0 to 59)			Second (0 to 59)		
n+3	Reserved			Reserved	Weekday (1 to 7, 1=Sunday)	
n+4 [1)]	Reserved			Reserved		
n+5 [1)]	Reserved			Reserved		

1) The two data words must exist in the data area to ensure that the data format matches WinCC flexible and to avoid reading false information.

Note

Note that when you enter the year, values 80 to 99 result in years 1980 through 1999 and the values 0 to 29 result in the years 2000 through 2029.

Permitted data types

You can use the following data types when you configure the "Date/Time PLC" area pointer:

- Word
- UInt
- DTL

Use of the "DTL" data type

Use data type "DTL" with communication driver S7 1200. A tag of the "DTL" data type has a length of 12 bytes and saves information on date and time in a predefined structure.

The "DTL" data type has the following structure:

Byte	Component	Data type	Value range
0	Year	UINT	1970 to 2554
1			
2	Month	USINT	0 to 12
3	Day	USINT	1 to 31
4	Day of week	USINT	1(Sunday) to 7(Saturday) The weekday is not considered in the value entry.
5	Hour	USINT	0 to 23
6	Minute	USINT	0 to 59
7	Second	USINT	0 to 59
8	Nanoseconds	UDINT	0 to 999 999 999
9			
10			
11			

The HMI devices do not support the use of nanoseconds. Values in the nanosecond range will be ignored during processing in Runtime.

Area pointer "Coordination"

Function

The "Coordination" area pointer is used to implement the following functions:

- Detecting the startup of the HMI device in the control program
- Detecting the current operating mode of the HMI device in the control program
- Detecting whether the HMI device is ready to communicate in the control program

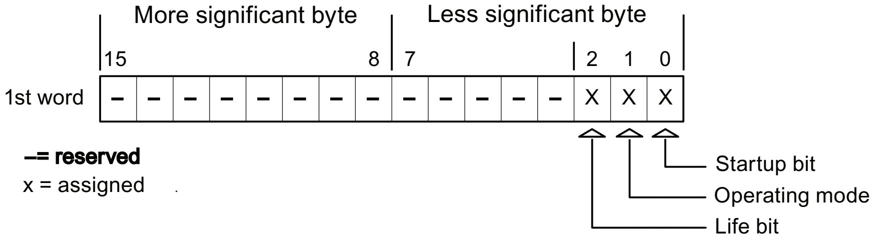
The "Coordination" area pointer has a length of one word.

Application

Note

Each time the area pointer is updated by the HMI device, the entire coordination area is always written.
For this reason, the PLC program must not make any changes in the coordination area.

Assignment of the bits in the "Coordination" area pointer



Startup bit

The startup bit is set briefly to "0" by the HMI device during startup. After startup, the bit is set permanently to "1".

Operating mode

As soon as the HMI device is switched offline by the user, the operating mode bit is set to 1. In normal operation of the HMI device, the state of the operating mode bit is "0". You can find out the current operating mode of the HMI device by querying this bit.

Life bit

The life bit is inverted by the HMI device at intervals of approximately one second. By querying this bit in the PLC program, you can check whether or not the connection to the HMI device still exists.

Processing in the PLC

For a simpler evaluation in the PLC program, use a Bool array for this area pointer when using the SIMATIC S7 1200 communication driver. You will have to map the complete 16-bit word of the area pointer. Configure a tag of the data type "Array [0 .. 15] of bool" for this purpose.

Permitted data types

You can use the following data types when you configure the "Coordination" area pointer.

- Word
- UInt
- Bool

Area pointer "Screen number"

Function

The HMI devices store information about the screen called up on the HMI device in the "Screen number" area pointer.

This allows the transfer of the current screen contents from the HMI device to the PLC. Certain reactions can be triggered in the PLC, such as the call of a different screen.

Use

Before the "Screen number" area pointer can be used, it must be set up and activated by selecting "Communication ▸ Area pointer". You can create only **one** instance of the "Screen number" area pointer and only on **one** PLC.

The screen number is always transferred to the PLC when a new screen is activated or when the focus within a screen changes from one screen object to another.

Structure

The area pointer is a data area in the memory of the PLC with a fixed length of 5 words.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1. Word	Current screen type															
2. Word	Current screen number															
3. Word	Reserved															
4th word	Current field number															
5. Word	Reserved															

- Current screen type

  "1" for root screen or
  "4" for permanent window
- Current screen number

  1 to 32767
- Current field number

  1 to 32767

Note

Device dependency

Permanent windows are not available on Basic Panels.

Permitted data types

You can use the following data types when you configure the "Screen number" area pointer.

- Word
- UInt

Area pointer "Project ID"

Function

When runtime starts it can check to see if the HMI device is connected to the correct PLC. This check is important when operating with several HMI devices.

The HMI device compares a value stored on the PLC with the value specified in the configuration data. This ensures the compatibility of the configuration data and the PLC program.

A missing compatibility results in a corresponding alarm and Runtime will not be started.

Use

In order to use this area pointer you must set up the following during the configuration:

- Define the version of configuration. Possible values between 1 and 255.

  You enter the version in the editor "Runtime settings > General" in the "Identification" area.
- This is where you select the PLC tag or the tag array that you have configured as the data area for the area pointer.

Connection failure

A connection failure to a device on which the "project ID" area pointer is configured results in all the other connections in the project being switched to "offline".

This behavior has the following prerequisites:

- You have several connections configured in a project.
- You are using the "project ID" area pointer in at least one connection.

Causes which may set connections "offline":

- The PLC is not available.
- The connection has been switched offline in the engineering system.

Permitted data types

You can use the following data types when you configure the "Project ID" area pointer.

- Word
- UInt

Area pointer "Job mailbox"

Function

The PLC can use the job mailbox to transfer jobs to the HMI device to trigger corresponding actions on the HMI device. These functions include, for example:

- Display screen
- Set date and time

Data structure

The first word of the job mailbox contains the job number. Depending on the job mailbox, up to three parameters can be transferred.

Word	Most significant byte	Least significant byte
n+0	0	Job number
n+1	Parameter 1	
n+2	Parameter 2	
n+3	Parameter 3	

The HMI device evaluates the job mailbox if the first word of this job is not equal to zero. This means that the parameters must be entered in the job mailbox first, followed by the job number.

When the HMI device accepts the job mailbox, the first word is set to 0 again. The execution of the job mailbox is generally not completed at this point in time.

Job mailboxes

All job mailboxes and their parameters are listed below. The "No." column contains the job number of the job mailbox. Job mailboxes can only be triggered by the PLC when the HMI device is online.

No.	Function	
14	**Set time (BCD-coded)**	
	Parameter 1	Left byte: - Right byte: hours (0-23)
	Parameter 2	Left byte: minutes (0-59) Right byte: seconds (0-59)

No.	Function	
14	Set time (BCD-coded)	
	Parameter 3	-
15	Set date (BCD-coded)	
	Parameter 1	Left byte: - Right byte: weekday (1-7: Sunday-Saturday)
	Parameter 2	Left byte: day (1-31) Right byte: month (1-12)
	Parameter 3	Left byte: year
23	User logon	
	Logs the user on with the name "PLC user" at the HMI device with the group number transferred in parameter 1. The logon is possible only when the transferred group number exists in the project.	
	Parameter 1	Group number 1 to 255
	Parameter 2, 3	-
24	User logoff	
	Logs off the current user. (The function corresponds to the "logoff" system function)	
	Parameter 1, 2, 3	-
40	Transferring date/time to PLC	
	An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
41	Transfer date/time to the PLC	
	An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
46	Updating tags	
	Causes the HMI device to read the current value of the tags from the PLC whose update ID matches the value transferred in parameter 1. (Function corresponds to the "UpdateTag" system function.)	
	Parameter 1	1 - 100
49	Clear event buffer	
	Deletes all analog alarms and discrete alarms of the "Warnings" class from the alarm buffer.	
	Parameter 1, 2, 3	-
50	Clear error alarm buffer	
	Deletes all analog alarms and discrete alarms of the "Errors" class from the alarm buffer.	
	Parameter 1, 2, 3	-
51	Display selection	
	Parameter 1	Screen number
	Parameter 2	-
	Parameter 3	Field number
69	Reading data record from PLC [1)]	

No.	Function	
14	Set time (BCD-coded)	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	0: Do not overwrite existing data record 1: Overwrite existing data record
70	Writing data record from PLC [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	-

[1]	Only devices supporting recipes
[2]	OP 73, OP 77A and TP 177A HMI devices also execute the "Screen selection" job mailbox if the on-screen keyboard is active.
[3]	The weekday is ignored on HMI device KTP 600 BASIC PN.

Permitted data types

You can use the following data types when you configure the "Screen number" area pointer.

- Word
- UInt

"Data record" area pointer

"Data mailbox" area pointer

Function

When data records are transferred between the HMI device and PLC, both partners access common communications areas on the PLC.

Data transfer types

There are two ways of transferring data records between the HMI device and PLC:

- Transfer without synchronization
- Transfer with synchronization over the data mailbox

Data records are always transferred directly. That is, the tag values are read from an address or written to an address configured for this tag directly, without redirecting the values by means of interim memory.

Initiating the transfer of data records

There are three ways of triggering the transfer:

- Operator input in the recipe view
- Job mailboxes

  The transfer of data records can also be triggered by the PLC.
- Triggering by configured functions

If the transfer of data records is triggered by a job mailbox, the data in the recipe view will be updated as well. Avoid operating the recipe view while job mailboxes for transfer of data records are being triggered. If you have already started editing a data record and a job mailbox is triggered for transfer of data records, then this job mailbox will be rejected.

Permitted data types

You can use the following data types when you configure the "Data record" area pointer.

- Word
- UInt

Transfer without synchronization

If you select asynchronous transfer of data records between the HMI device and PLC, there is no coordination over the common data areas. It is therefore unnecessary to set up a data area during configuration.

Asynchronous data record transfer can be a useful alternative, for example, when:

- The system is capable of excluding the risk of uncontrolled overwriting of data by the communication peer.
- The PLC does not require information about the recipe number and data record number.
- The transfer of data records is triggered by the operator of the HMI device.

Reading values

When a read job is triggered, the values are read from the PLC addresses and transferred to the HMI device.

- Triggering by the operator in the recipe view:

  The values are downloaded to the HMI device. You can then process, edit, or save these values, for example.
- Triggering by a function or job mailbox:

  The values are saved immediately to the data volume.

Writing values

When a write job is triggered, the values are written to the PLC addresses.

- Triggering by the operator in the recipe view:

  The current values are written to the PLC.
- Triggering by a function or job mailbox:

  The current values are written to the PLC from the data medium.

Sequence of a transfer started by the operator in the recipe display

Reading from the PLC started by the operator in the recipe view

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe number to be read and the status "Transferring" in the data mailbox and sets the data record number to 0.	Abort with system alarm.
3	The HMI device reads the values from the PLC and displays them in the recipe view. If the recipes have synchronized tags, the values from the PLC are also written to the tags.	
4	The HMI device sets the status "Transfer completed."	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC started by the operator in the recipe view

Step	Action	
	Check: Status word = 0?	
1	Yes	No
	The HMI device enters the recipe and data record number to be written and the status "Transferring" in the data mailbox.	Abort with system alarm.
2	The HMI device writes the current values to the PLC. If the recipes have synchronized tags, the changed values are synchronized between the recipe view and tags and then written to the PLC.	
3	The HMI device sets the status "Transfer completed."	
4	If required, the control program can now evaluate the transferred data.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Sequence of the transfer triggered by a job mailbox

The transfer of data records between the HMI device and the PLC can be initiated by either one of these stations.

The two job mailboxes No. 69 and No. 70 are available for this type of transfer.

No. 69: Read data record from PLC ("PLC → DAT")

Job mailbox no. 69 transfers data mailboxes from the PLC to the HMI device. The job mailbox is structured as follows:

	Most significant byte	Least significant byte
Word 1	0	69
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	Do not overwrite existing data record: 0 Overwrite existing data record: 1	

No. 70: Write data record to PLC ("DAT → PLC")

Job mailbox no. 70 transfers data mailboxes from the HMI device to the PLC. The job mailbox is structured as follows:

	Most significant byte	Least significant byte
Word 1	0	70
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	—	

Sequence when reading from the PLC with job mailbox "PLC → DAT" (no. 69)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device reads the values and stores the values in the data record specified in the job mailbox.	
4	• If "Overwrite" was selected in the job, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed". • If "Do not overwrite" was selected in the job, and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Sequence writing to the PLC with job mailbox "DAT → PLC" (no. 70)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device fetches the values of the data record specified in the job from the data medium and writes the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The PLC program can now evaluate the transferred data. To allow further transfers, the PLC program must set the status word to 0 again.	

Sequence of the transfer when triggered by a configured function

Reading from the PLC using a configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system alarm.

Step	Action	
3	The HMI device reads the values from the PLC and stores them in the data record specified in the function.	
4	• If "Yes" was selected for the "Overwrite" function, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "No" was selected for the "Overwrite" function and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC by means of configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system alarm.
3	The HMI device fetches the values of the data record specified in the function from the data medium and transfers the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Possible causes of error when transferring data records

Possible causes of error

The section below shows possible error causes which lead to the cancellation of data record transfer:

- Tag address not set up on the PLC
- Overwriting data records not possible
- Recipe number does not exist
- Data record number does not exist

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Reaction to an aborted transfer due to errors

If the transfer of data records is aborted due to errors, the HMI device reacts as follows:

- Triggering by the operator in the recipe view

  Information in the status bar of the recipe view and output of system alarms
- Triggered by function

  Output of system alarms
- Triggering by job mailbox

  No return message on the HMI device

You can nonetheless evaluate the status of the transfer by querying the status word in the data mailbox.

Note

Availability for specific devices

Notes in the status bar of the recipe view are not available in Basic Panels.

Trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out time-triggered for Basic Panels.

For additional information see:

Configuring trend displays for values from the PLC (Page 2155)

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration.

Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Alarms

Configuring alarms

Configure alarms

Several steps are needed to configure alarms, such as operational messages, error alarms, and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with alarms (Page 2167)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a SIMATIC communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
SIMATIC S7 PLCs	WORD, INT	BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, TIMER

How the bit positions are counted

For connections with a SIMATIC communication driver, the following counting method applies:

How the bit positions are counted	Byte 0								Byte 1							
	Most significant byte								Least significant byte							
In SIMATIC S7 PLCs	7							0	7							0
In WinCC you configure:	15							8	7							0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

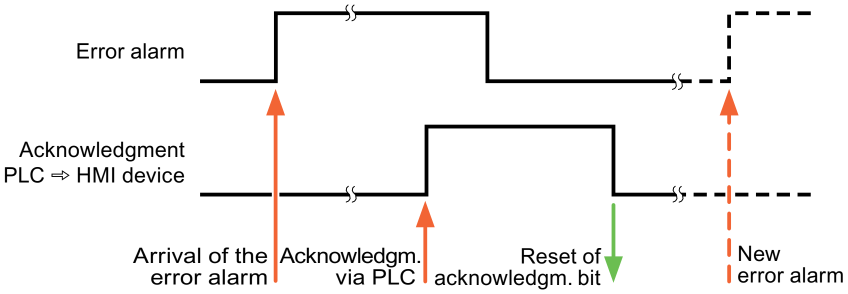
Acknowledgment by the PLC

In "Write acknowledgment tag", you configure the tag or the array tag and the bit number based on which the HMI device can recognize an acknowledgment by the PLC.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In "Read acknowledgment tag", you configure the tag or the array tag and the bit number that is written to the PLC after acknowledgment from the HMI device. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

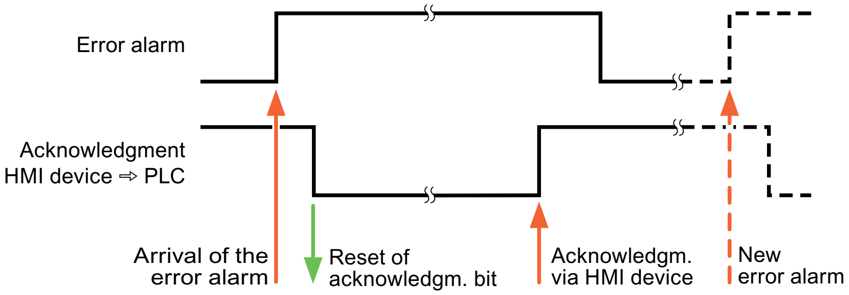
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



10.8.5.5 Performance features of communication

Permitted data types for SIMATIC S7 1200 - V1

Permitted data types for connections with SIMATIC S7 1200 (V1)

V1: Firmware Version V1.0

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Length
BOOL	1 bit
SINT	1 byte
INT	2 bytes
DINT	4 bytes
USINT	1 byte
UINT	2 bytes
UDINT	4 bytes
REAL	4 bytes
LREAL	8 bytes
TIME	4 bytes
DTL	12 bytes
STRING	(2+n) bytes, n = 0 to 254
CHAR	1 byte
Array of CHAR	--
BYTE	1 byte
WORD	2 bytes
DWORD	4 bytes

Permitted data types for SIMATIC S7 1200 - V2

Permitted data types for connections with SIMATIC S7 1200 (V2)

V2: Firmware Version V2.0

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Length
BOOL	1 bit
SINT	1 byte
INT	2 bytes
DINT	4 bytes
USINT	1 byte
UINT	2 bytes
UDINT	4 bytes
REAL	4 bytes
LREAL	8 bytes
TIME	4 bytes
DATE	2 bytes
DTL	12 bytes
TIME_OF_DAY, TOD	4 bytes
STRING	(2+n) bytes, n = 0 to 254
CHAR	1 byte
Array of CHAR	--
BYTE	1 byte
WORD	2 bytes
DWORD	4 bytes

10.8.5.6 Creating connections in the "Connections" editor

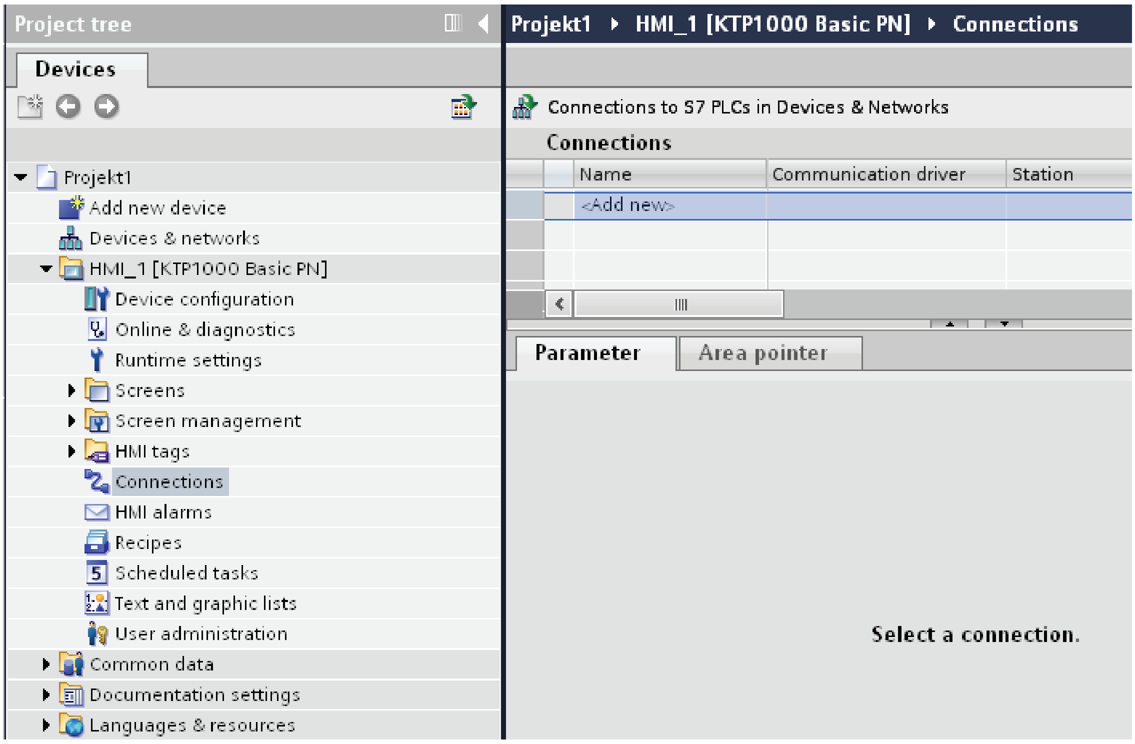
Creating a PROFINET connection

Requirements

- A project is open.
- An HMI device with a PROFINET interface has been created.

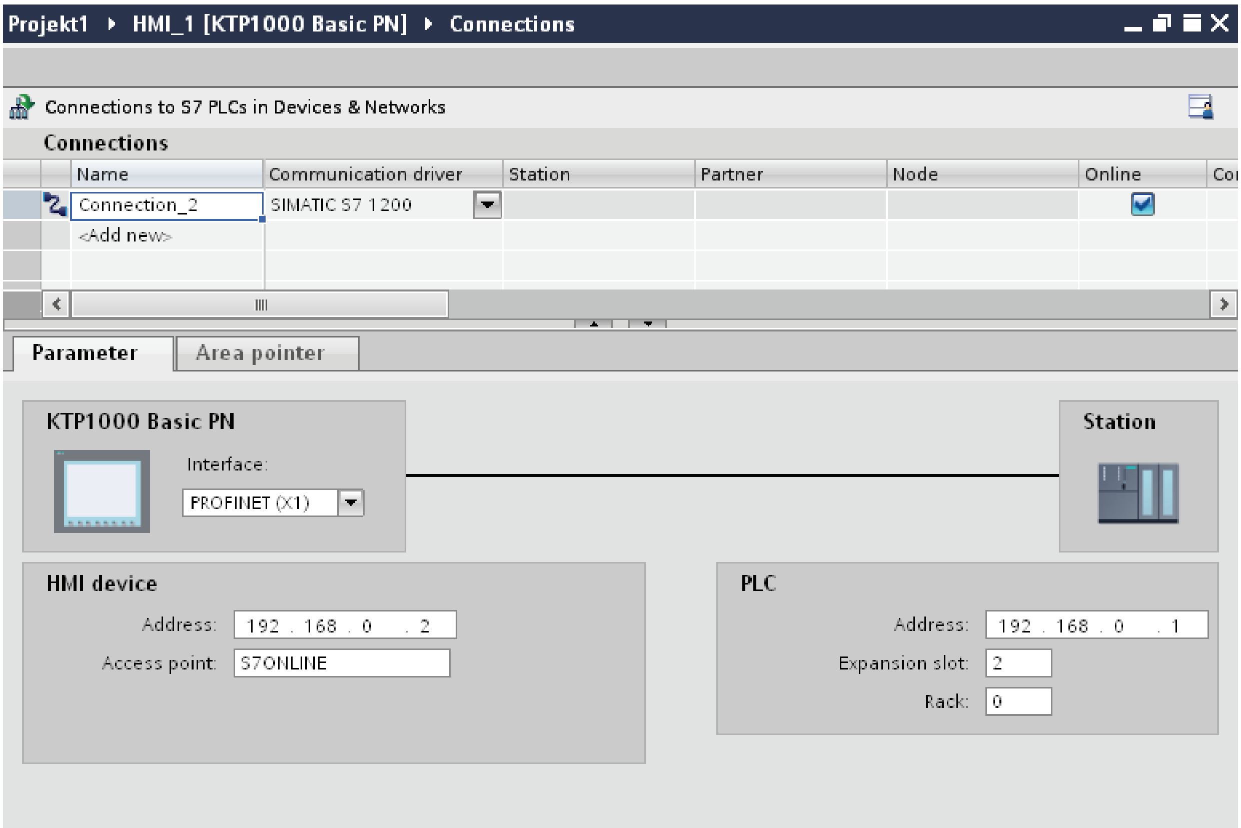
Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



Name	Communication driver	Station
<Add new>		

3. In the "Communication drivers" column, select the "SIMATIC S7 1200" driver.
4. Click the name of the connection.
5. Select a PROFINET interface of the HMI device in the Inspector window under "Parameters > Interface".


Projekt1 ▸ HMI_1 [KTP1000 Basic PN] ▸ Connections
Connections to S7 PLCs in Devices & Networks
Connections
Parameter
Area pointer
KTP1000 Basic PN
Station
HMI device
PLC

Name	Communication driver	Station	Partner	Node	Online	Co
Connection_2	SIMATIC S7 1200				☑	
<Add new>						

Interface: PROFINET (X1)

HMI device

Address: 192 . 168 . 0 . 2

Access point: S7ONLINE

PLC

Address: 192 . 168 . 0 . 1

Expansion slot: 2

Rack: 0

6. Set the IP addresses of the communication partners in the Inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

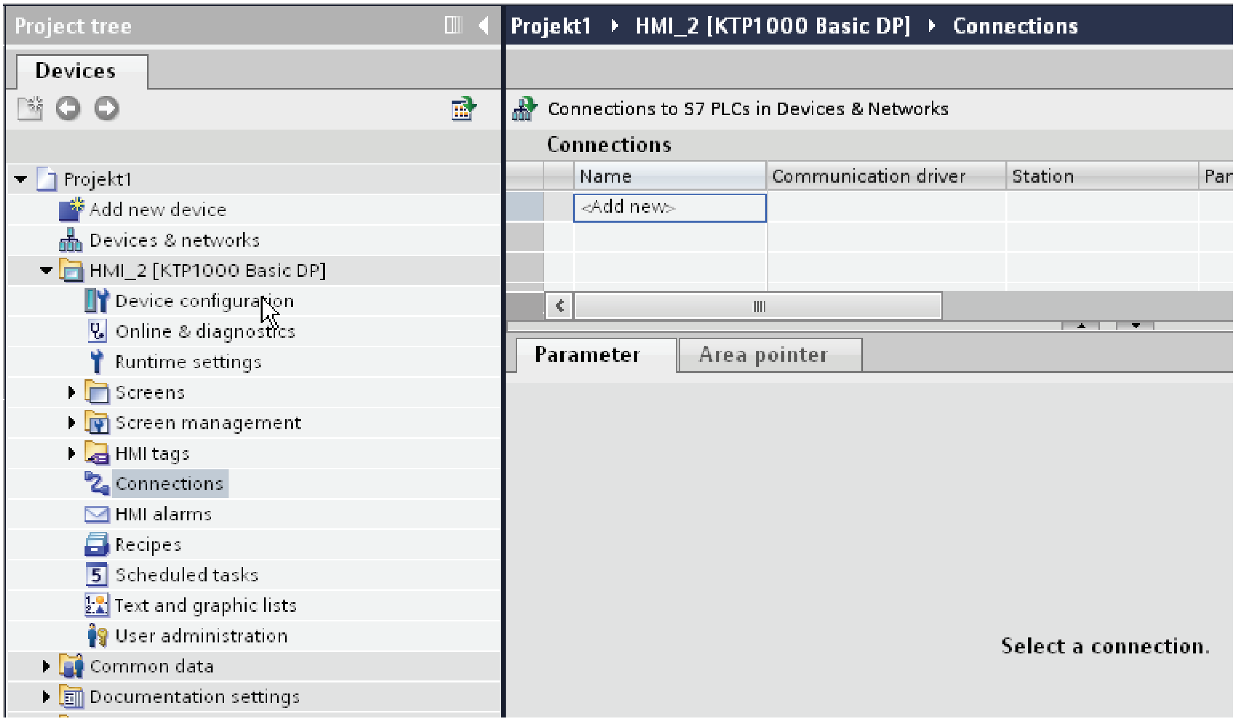
Creating a PROFIBUS DP connection

Requirements

- A project is open.
- An HMI device with a PROFIBUS interface has been created.

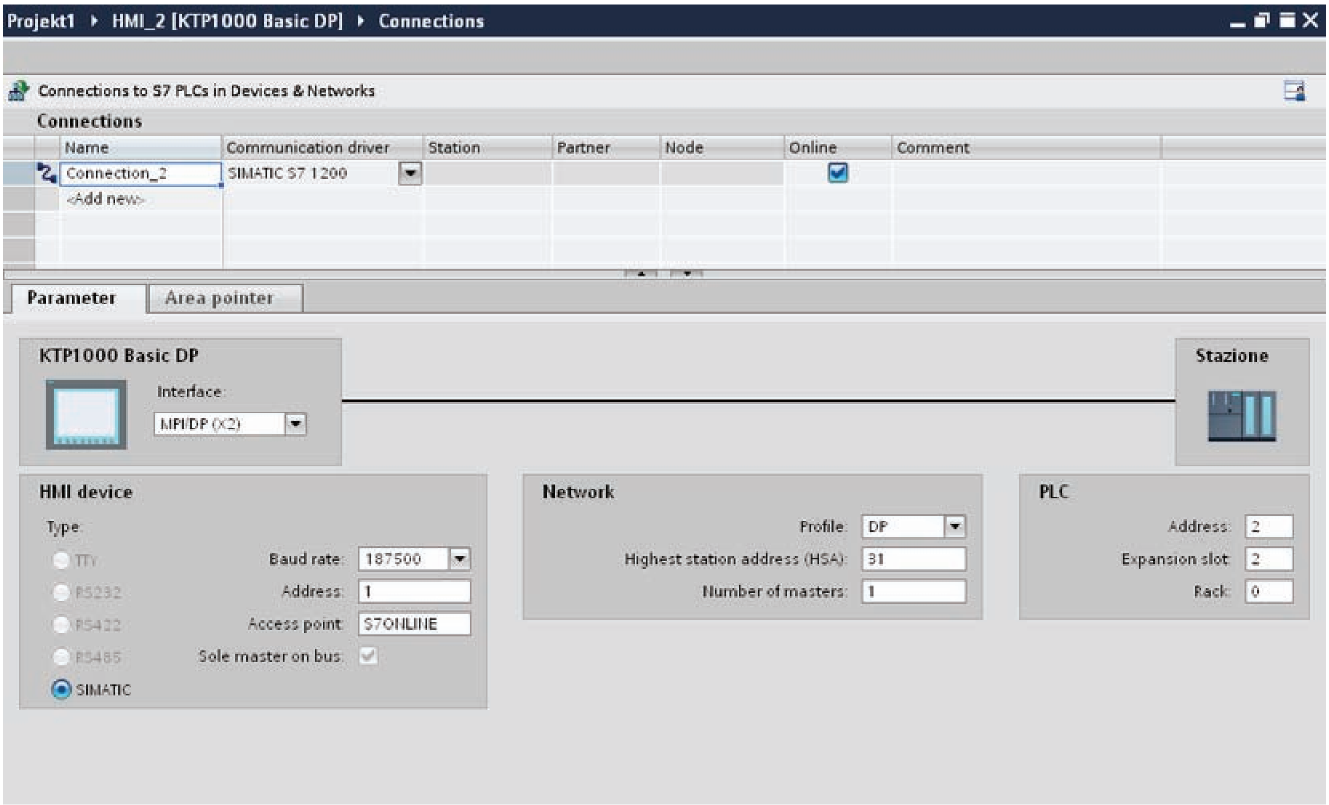
Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



Select a connection.

3. In the "Communication drivers" column, select the "SIMATIC S7 1200" driver.
4. Click the name of the connection.
5. Select the "MPI/DP" interface in the Inspector window under "Parameters".
6. Select the "DP" profile in the Inspector window under "Parameters > Network".



Projekt1 ▸ HMI_2 [KTP1000 Basic DP] ▸ Connections

Connections to S7 PLCs in Devices & Networks

Connections

Name	Communication driver	Station	Partner	Node	Online	Comment
Connection_2	SIMATIC S7 1200				☑	
<Add new>						

Parameter | Area pointer

KTP1000 Basic DP

Interface: MPI/DP (X2)

Stazione

HMI device

Type:

○ TTY
○ RS232
○ RS422
○ RS485
● SIMATIC

Baud rate: 187500
Address: 1
Access point: S7ONLINE
Sole master on bus: ☑

Network

Profile: DP
Highest station address (HSA): 31
Number of masters: 1

PLC

Address: 2
Expansion slot: 2
Rack: 0

7. Set the addresses of the communication partners in the Inspector window:
   – HMI device: "Parameters > HMI device > Address"
   – PLC: "Parameters > PLC > Address"

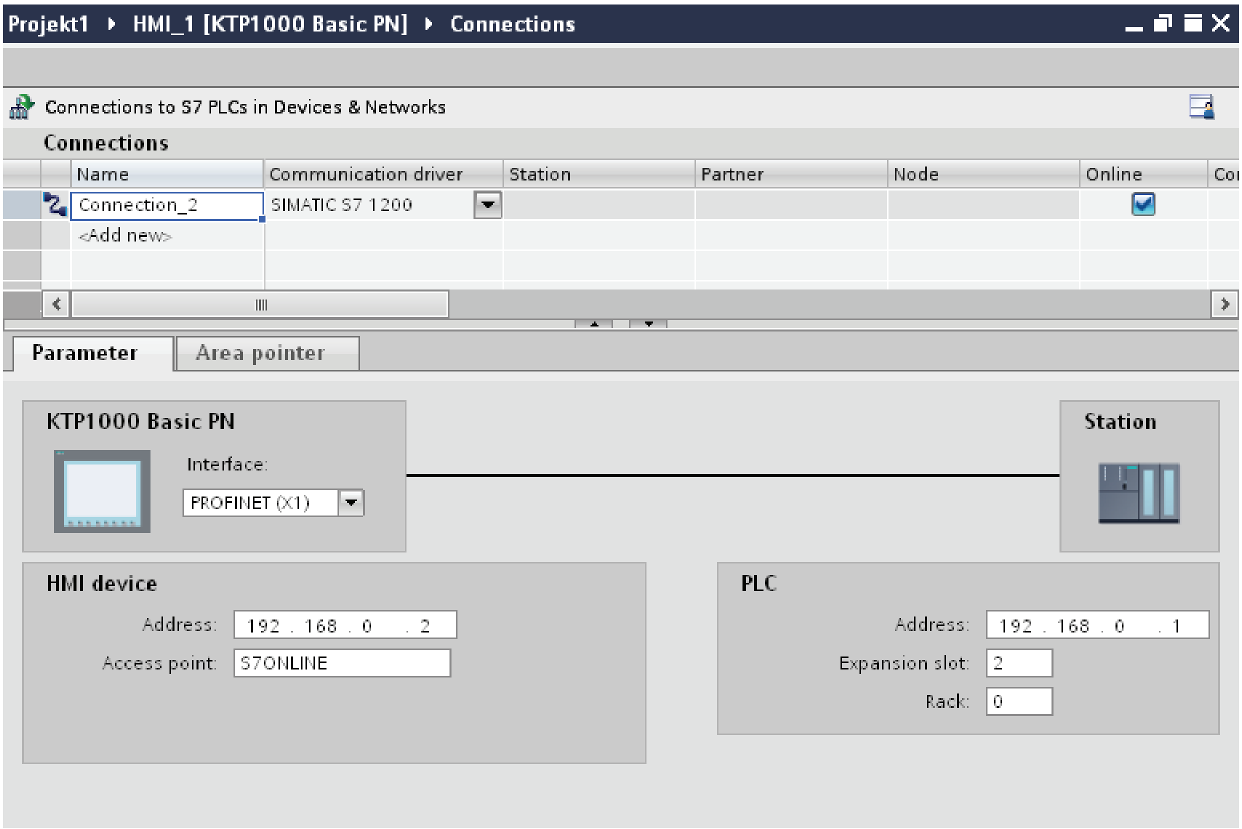
Parameters for the connection

Parameters for the connection (SIMATIC S7 1200)

Parameters to be set

To assign the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device", "Network", and "PLC" areas are available for assigning parameters according to the interface used.



Projekt1 ▸ HMI_1 [KTP1000 Basic PN] ▸ Connections

Connections to S7 PLCs in Devices & Networks

Connections

Name	Communication driver	Station	Partner	Node	Online	Co
Connection_2	SIMATIC S7 1200				☑	
<Add new>						

Parameter | Area pointer

KTP1000 Basic PN

Interface: PROFINET (X1)

Station

HMI device

Address: 192 . 168 . 0 . 2

Access point: S7ONLINE

PLC

Address: 192 . 168 . 0 . 1

Expansion slot: 2

Rack: 0

Ethernet parameters

Parameters for the HMI device

You set the parameters for the HMI device in the network under "HMI device".. The changes are not transferred automatically to the HMI device. You must change the settings in the Control Panel of the HMI device.

- "Interface"

  If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC.

Note

The IP address in the Control Panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

The IP address already set up in the Control Panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

The IP address is transferred to the HMI device during project transfer.

To set up the IP address of the HMI device:

- Click the HMI device.
- Open the "Device configuration" editor.
- Click the Ethernet interface.
- Assign the IP address in the inspector window under:

  "General > PROFINET interface > Ethernet addresses"

- "Address"

  You assign the IP address of the HMI device in the "Address" area.

  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.

- "Access point"

  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"

  Under "Address", set the IP address of the S7 module to which the HMI device is connected.

- "Expansion slot"

  Defines the number of the expansion slot of the CPU to be addressed.

- "Rack"

  Defines the rack number of the CPU to be addressed.

- "Cyclic mode"

  Note

  The setting "Cyclic operation" cannot be configured for the SIMATIC S7 1200 PLC.

  When cyclic operation is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This increases system performance.

  Disable cyclic mode if you are operating several HMI devices in parallel.

PROFIBUS parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"

  Specifies the physical connection used.

- "Interface"

  For "Interface", you select the HMI device interface via which the HMI device is connected to the PROFIBUS network.

- "Baud rate"

  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"

  You set the PROFIBUS address of the HMI device under "Address". The PROFIBUS address must be unique in the PROFIBUS network.

- "Sole master on bus"

  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master).

  In S7 200, you must set an HMI device as the master.

- "Access point"

  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the network

Under "Network", you set the parameters for the PROFIBUS network to which the HMI device is linked.

- "Profile"

  For "Profile", you select the network profile that is used in the network. In "Profile", set "DP", "Universal", or "Standard". The setting must be identical throughout the network.

- "Highest address"

  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual PROFIBUS address. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud in the OP 73 or the OP 77A, the highest station address must be less than or equal 63.

- "Number of masters"

  For "Number of masters", set the number of masters in the PROFIBUS network. This information is necessary to correctly calculate the bus parameters.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"

  For "Address", set the PROFIBUS address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.

- "Cyclic operation"

  Note

  The setting "Cyclic operation" cannot be configured for the SIMATIC S7 1200 PLC.

  When cyclic operation is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This increases system performance. Disable cyclic mode if you are operating several HMI devices in parallel. This setting is not required for SIMATIC S7-200 PLCs.

10.8.6 Communicating with SIMATIC S7 300/400

10.8.6.1 Communication with SIMATIC S7 300/400

Introduction

This section describes the communication between an HMI device and the SIMATIC S7 300 and S7 400 PLCs. These two PLCs will be referred to jointly as SIMATIC S7 300/400.

You can configure the following communication channels for the SIMATIC S7 300/400 PLC:

- PROFINET
- PROFIBUS
- MPI

HMI connection for communication

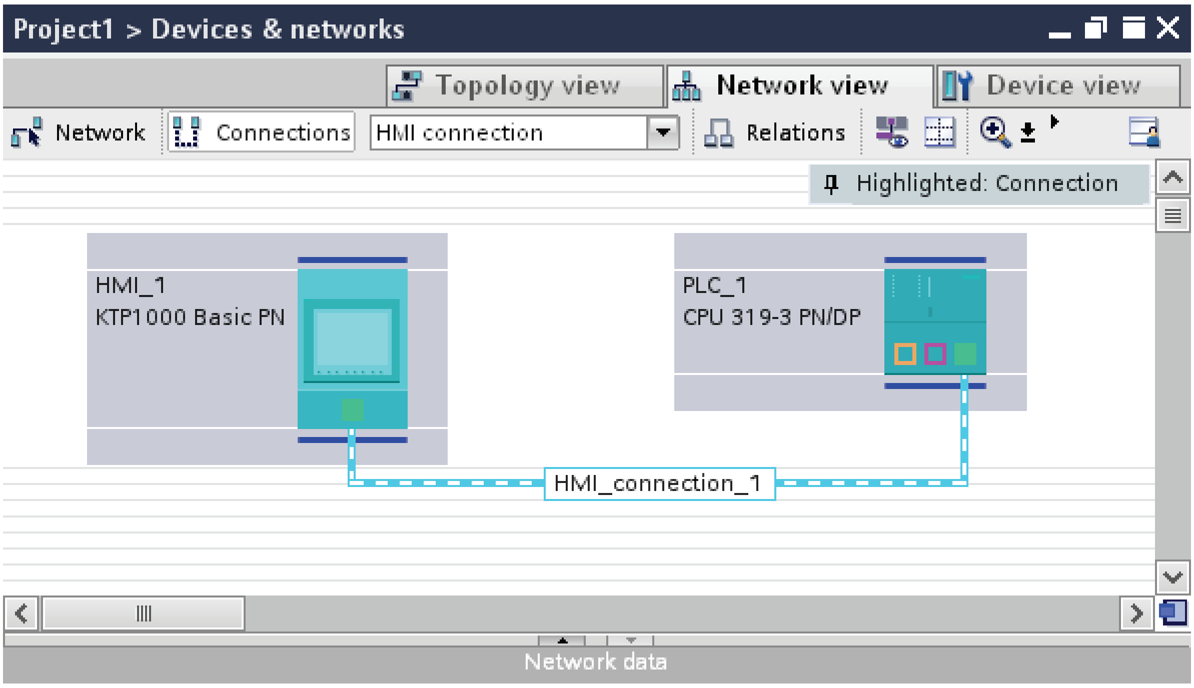
You configure connections between the HMI device and a SIMATIC S7 300/400 in the "Devices & Networks" editor.

10.8.6.2 Communication via PROFINET

Communication via PROFINET

HMI connections via PROFINET

If you have inserted an HMI device and a SIMATIC S7 300/400 into the project, you interconnect the two PROFINET interfaces in the "Devices & Networks" editor.



You can also connect multiple HMI devices to one SIMATIC S7 300/400 and multiple SIMATIC S7 300/400s to one HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFINET in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFINET in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFINET

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 300/400 via PROFINET or Ethernet in the "Devices & Networks" editor.

⚠ CAUTION

Communication via Ethernet

In Ethernet-based communication, the end user is responsible for the security of his data network.

Targeted attacks can overload the device and interfere with proper functioning.

Requirements

The following communication partners are created in the "Devices & Networks" editor:

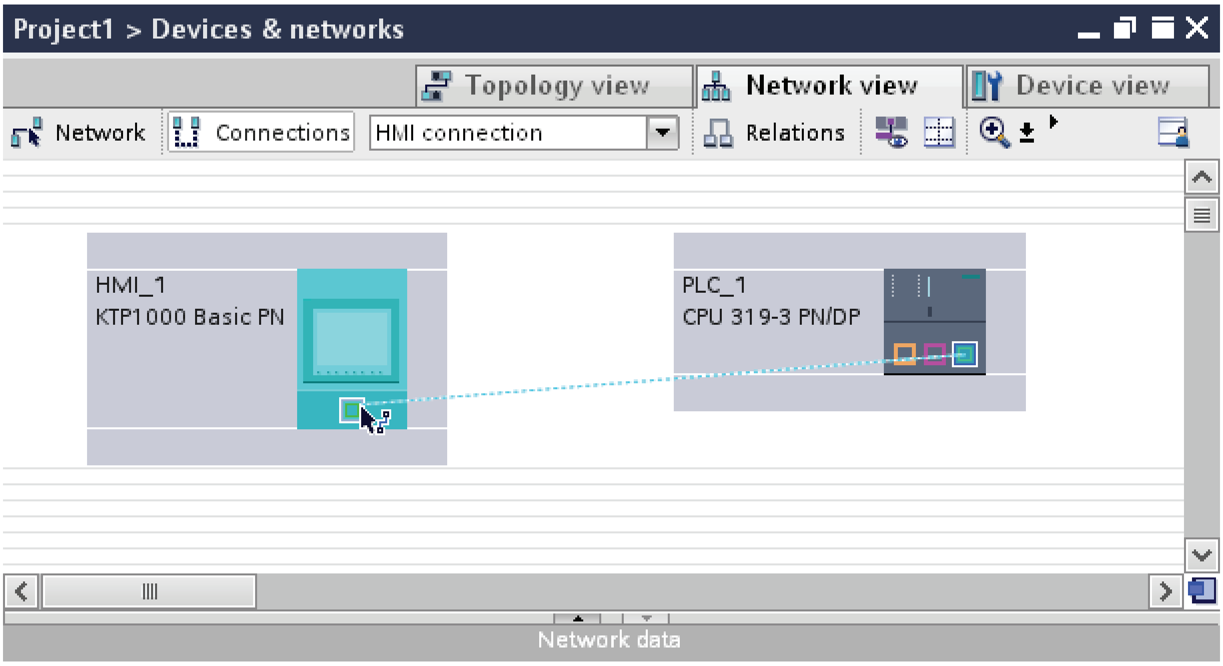
- HMI device with PROFINET or Ethernet interface
- SIMATIC S7 300/400 with PROFINET interface.

Procedure

1. Double-click the "Devices & Networks" item in the project tree.

   The available communication partners in the project are displayed graphically in the network view.
2. Click the "Connections" button and select "HMI connection" for the connection type.

   The devices available for connection are highlighted in color.
3. Click the PROFINET interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFINET or Ethernet interface of the HMI device.



4. Click the connecting line.
5. Click "Highlight HMI connection" and select the HMI connection.

   The connection is displayed graphically in the Inspector window.
6. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.

   See the chapter "PROFINET parameters (Page 2474)" for additional details.

   Note

   The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

   You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC S7 300/400. The IP address and subnet mask connection parameters are configured.

PROFINET parameters

PROFINET parameters for the HMI connection

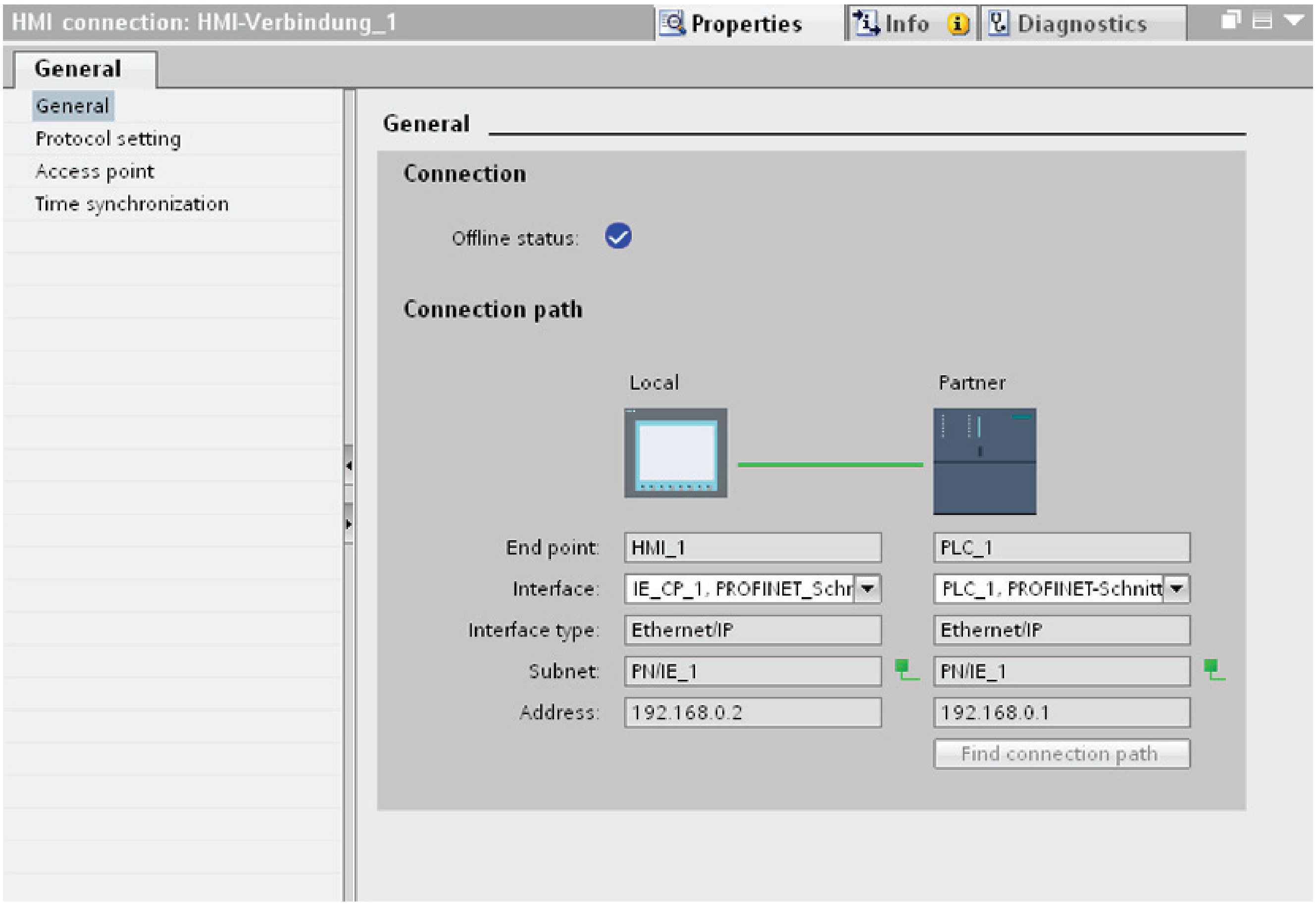
PROFINET parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and changing the HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



Offline status: ✓

	Local	Partner
End point:	HMI_1	PLC_1
Interface:	IE_CP_1, PROFINET_Schr	PLC_1, PROFINET-Schnitt
Interface type:	Ethernet/IP	Ethernet/IP
Subnet:	PN/IE_1	PN/IE_1
Address:	192.168.0.2	192.168.0.1

"Connection"

Displays whether the devices are networked together.

✓ - displayed if the devices are networked together.

❗ - displayed if the devices are not networked together.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFINET parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"

  Displays the device name. This area cannot be edited.
- "Interface"

  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"

  Displays the selected interface type. This area cannot be edited.
- "Subnet"

  Displays the selected subnet. This area cannot be edited.
- "Address"

  Displays the selected IP address of the device. This area cannot be edited.
- "Find connection path" button

  Enables the subsequent specification of connections.

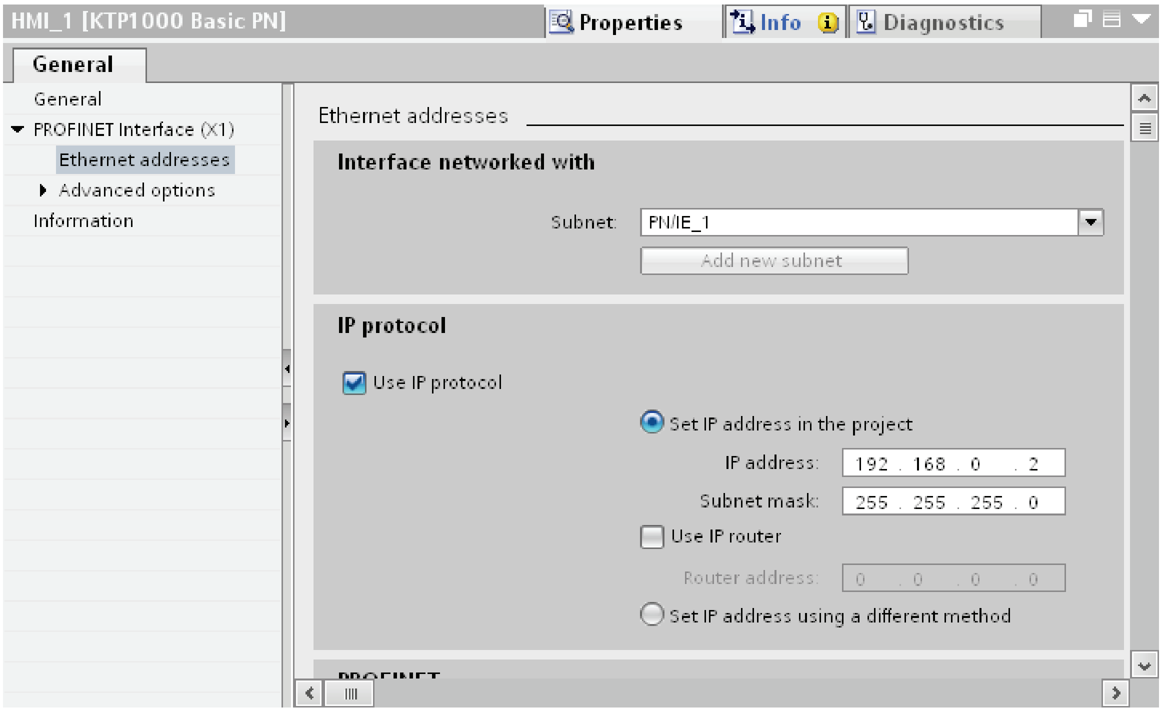
PROFINET parameters for the HMI device

PROFINET parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFINET parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General".



Subnet: PN/IE_1

☑ Use IP protocol

● Set IP address in the project

IP address: 192 . 168 . 0 . 2

Subnet mask: 255 . 255 . 255 . 0

□ Use IP router

Router address: 0 . 0 . 0 . 0

○ Set IP address using a different method

"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Set IP address in the project"

  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.

  Note

  The device is automatically restarted in the case of HMI devices with the Windows CE 3.0 operating system.

  HMI devices with Windows CE 3.0:

  - OP 77B
  - TP 177B color PN/DP
  - TP 177B mono DP
  - OP 177B color PN/DP
  - OP 177B mono DP
  - Mobile Panel 177 PN
  - Mobile Panel 177 DP
  - TP 277 6"
  - OP 277 6"

- "Subnet mask"

  You assign data of the subnet mask in the "Subnet mask" area.

- "Use IP router"

  If you are using an IP router, select "Use IP router" and enter the router address in the "Router address" field.

- "Set IP address using a different method"

  If the function "Set IP address using a different method" is activated, the IP address is not taken from the project. You have to enter the IP address directly in the Control Panel of the HMI device.

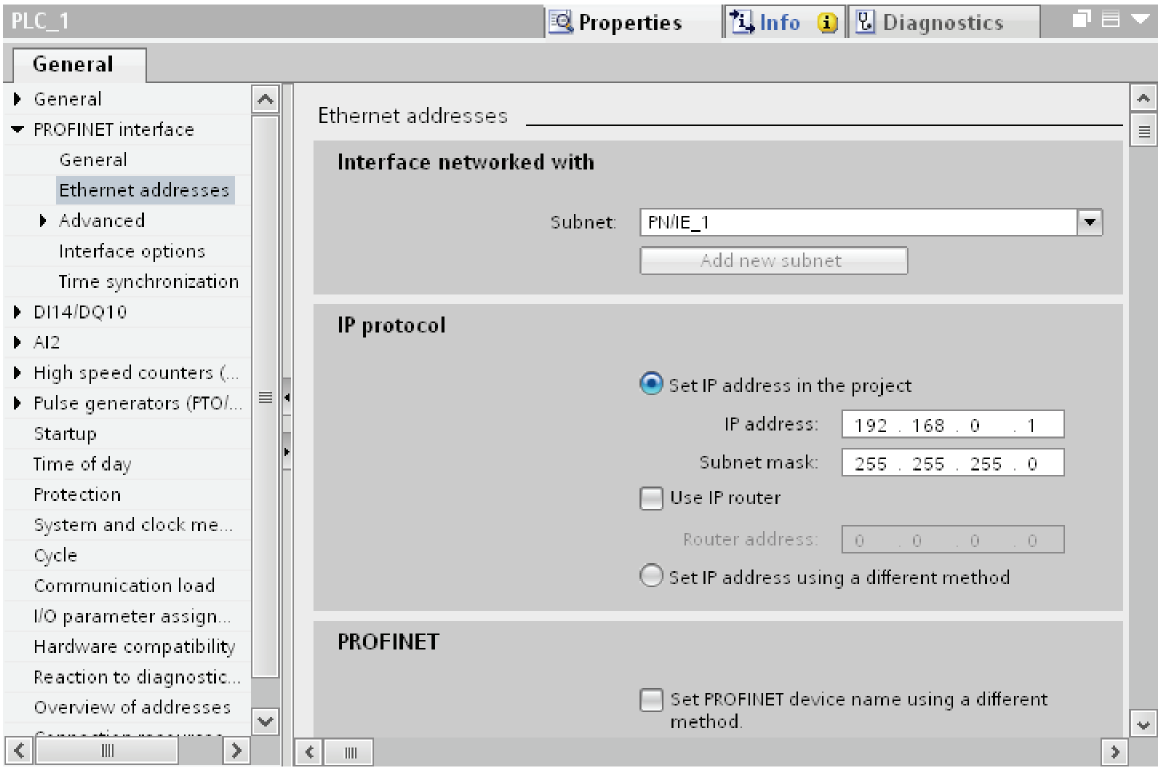
PROFINET parameters for the PLC

PROFINET parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFINET parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



Subnet: PN/IE_1

● Set IP address in the project

IP address: 192 . 168 . 0 . 1

Subnet mask: 255 . 255 . 255 . 0

□ Use IP router

Router address: 0 . 0 . 0 . 0

○ Set IP address using a different method

□ Set PROFINET device name using a different method.

"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Interface type"

  Depending on the HMI device type, you have various interfaces to choose from.
- "IP address"

  You assign the IP address of the HMI device in the "IP address" area.
- "Subnet mask"

  You assign data of the subnet mask in the "Subnet mask" area.

  If you are using an IP router, select "Use IP router" and enter the router address in the field.

Configuring Industrial Ethernet

Rules for the network configuration

The Ethernet interfaces of the devices have a default IP address that you can change.

IP address

The IP parameters are visible if the communication-capable devices support the TCP/IP protocol.

The IP address consists of 4 decimal figures in the range of 0 to 255. The decimal figures are separated from one another by a dot.

Example: 140.80.0.2

The IP address consists of the following:

- The address of the (sub) net
- The address of the node (generally also called host or network node)

Subnet mask

The subnet mask splits these two addresses. It determines which part of the IP address addresses the network and which part of the IP address addresses the node.

The set bits of the subnet mask determine the network part of the IP address.

Example:

Subnet mask: 255.255.0.0 = 11111111.11111111.00000000.00000000

In the example given for the above IP address, the subnet mask shown here has the following meaning:

The first 2 bytes of the IP address identify the subnet - i.e. 140.80. The last two bytes address the node, thus 0.2.

It is generally true that:

- The network address results from AND linking the IP address and subnet mask.
- The node address results from AND NOT linking the IP address and subnet mask.

Relation between IP address and default subnet mask

An agreement exists relating to the assignment of IP address ranges and so-called "Default subnet masks". The first decimal number (from the left) in the IP address determines the structure of the default subnet mask. It determines the number of "1" values (binary) as follows:

IP address (decimal)	IP address (binary)	Address class	Default subnet mask
0 to 126	0xxxxxxx.xxxxxxxx....	A	255.0.0.0
128 to 191	10xxxxxx.xxxxxxxx...	B	255.255.0.0
192 to 223	110xxxxx.xxxxxxxx...	C	255.255.255.0

Note

Range of values for the first decimal point

A value of between 224 and 255 is also possible for the first decimal number of the IP address (address class D etc). This is, however, not recommended because there is no address check for these values.

Masking other subnets

You can use the subnet mask to add further structures and form "private" subnets for a subnet that is assigned one of the address classes A, B or C. This is done by setting other lower points of the subnet mask to "1". For each bit set to "1", the number of "private" networks doubles and the number of nodes they contain is halved. Externally, the network functions like an individual network as it did previously.

Example:

You have a subnet of address class B (e.g. IP address 129.80.xxx.xxx) and change the default subnet mask as follows:

Masks	Decimal	Binary
Default subnet mask	255.255.0.0	11111111.11111111.00000000.00000000
Subnet mask	255.255.128.0	11111111.11111111.10000000.00000000

Result:

All nodes with addresses between 129.80.001.xxx and 129.80.127.xxx are on one subnet, all nodes with addresses between 129.80.128.xxx and 129.80.255.xxx are on another subnet.

Router

The job of the routers is to connect the subnets. If an IP datagram is to be sent to another network, it first has to be conveyed to a router. To make this possible, in this case you have to enter the address of the router for each node in the subnet.

The IP address of a node in the subnet and the address of the router may only differ at the points at which there is a "0" in the subnet mask.

Setting port options

Setting the port options

Changing connection settings for the PROFINET IO port

You can change the network settings for the PROFINET IO port as required. By default, the settings are made automatically. In normal situations, this guarantees problem-free communication.

Possible settings for transmission rate / duplex

Depending on the selected device, you can make the following settings for "Transmission rate / duplex":

- Automatic setting

  Recommended default setting of the port. The transmission settings are automatically "negotiated" with the peer port. The "Enable autonegotiation" option is also enabled as a default, in other words, you can use cross cables or patch cables for the connection.

- TP/ITP at x Mbps full duplex (half duplex)

  Setting of the transmission rate and the full duplex/half duplex mode. The effectiveness depends on the "Enable autonegotiation" setting:

  – Autonegotiation enabled

    You can use both cross cable and patch cable.

  – Autonegotiation disabled

    Make sure that you use the correct cable (patch cable or cross cable)! The port is also monitored with this setting.

- Deactivated

  Depending on the module type, the drop down list box can contain the "- Disabled -" option. This option, for example, allows you to prevent access to an unused port for security reasons. With this setting, diagnostic events are not generated.

"Monitor" option

This option enables or disables port diagnostics. Examples of port diagnostics: The link status is monitored, in other words, the diagnostics are generated during link-down and the system reserve is monitored in the case of fiber optic ports.

Option "Enable autonegotiation "

The autonegotiation setting can only be changed if a concrete medium (for example, TP 100 Mbps full duplex) is selected. Whether or not a concrete medium can be set depends on the properties of the module.

If autonegotiation is disabled, this causes the port to be permanently specified, as for example, is necessary for a prioritized startup of the IO device.

You must make sure the partner port has the same settings because with this option the operating parameters of the connected network are not detected and the data transmission rate and transmission mode can accordingly not be optimally set.

Note

When a local port is connected, STEP 7 makes the setting for the partner port if the partner port supports the setting. If the partner port does not accept the setting, an error message is generated.

Wiring rules for disabled autonegotiation

Requirements

You have made the following settings for the port in question, for example, to accelerate the startup time of the IO device:

- Fixed transmission rate
- Autonegotiation incl. autocrossing disabled

The time for negotiating the transmission rate during startup has been saved.

If you have disabled autonegotiation, you must observe the wiring rules.

Wiring rules for disabled autonegotiation

PROFINET devices have the following two types of ports:

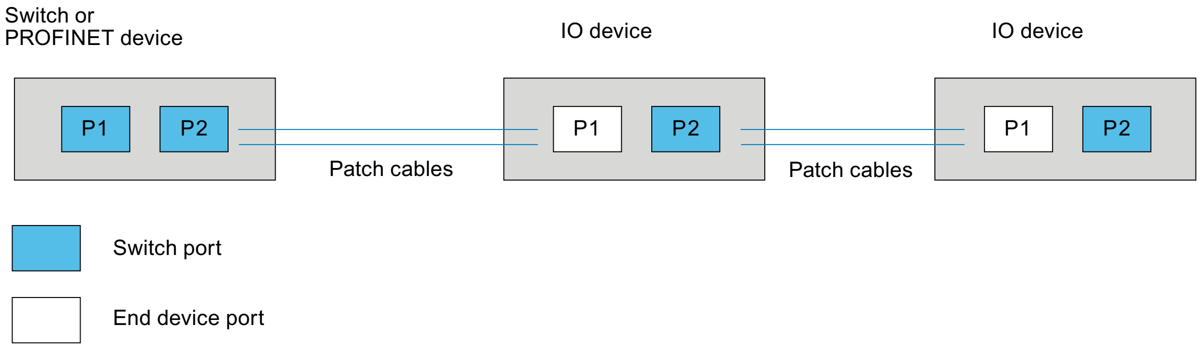
Type of port	PROFINET devices	Note
Switch port with crossed pin assignment	For IO devices: Port 2 For S7 CPUs with 2 ports: Ports 1 and 2	Crossed pin assignment means that the pin assignment for the ports for sending and receiving between the respective PROFINET devices is exchanged internally.
End device port with uncrossed pin assignment	For IO devices: Port 1 For S7 CPUs with one port: Port 1	-

Validity of the wiring rules

The cabling rules described in the following paragraph apply exclusively for the situation in which you have specified a fixed port setting.

Rules for cabling

You can connect several IO devices in line using a single cable type (patch cable). To do this, you connect port 2 of the IO device (distributed I/O) with port 1 of the next IO device. The following graphic gives an example with two IO devices.



Boundaries at the port

Requirements

To use boundaries, the respective device must have more than one port. If the PROFINET does not support boundary settings, they are not shown.

Enable boundaries

"Boundaries" are limits for transmission of certain Ethernet frames. The following boundaries can be set at a port:

- "End of discovery of accessible devices"

  No forwarding of DCP frames to identify accessible devices. Devices downstream from this port cannot be reached by the project tree under "Accessible devices". Devices downstream from this port cannot be reached by the CPU.

- "End of topology discovery"

  LLDP frames (Link Layer Discovery Protocol) are not forwarded for topology detection.

- "End of sync domain"

  No forwarding of sync frames transmitted to synchronize nodes within a sync domain.

  If you operate, for example, a PROFINET device with more than two ports in a ring, you should prevent the sync frame from being fed into the ring by setting a sync boundary (at the ports not inside the ring).

  Additional example: If you want to use several sync domains, configure a sync domain boundary for the port connected to a PROFINET device from the other sync domain.

Restrictions

The following restrictions must be observed:

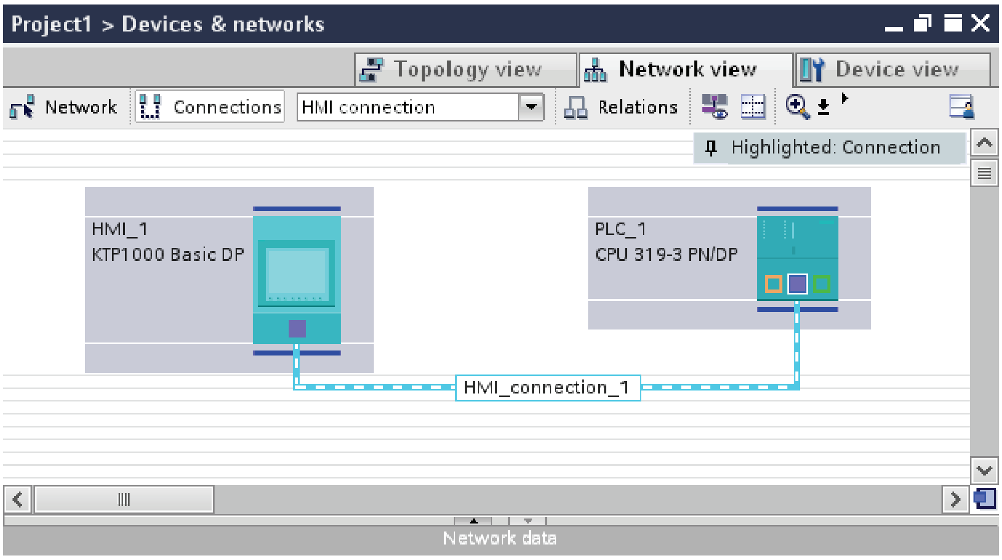
- The individual check boxes can only be used if the port supports the function in question.
- If a partner port has been determined for the port, the following check boxes cannot be used:
  - "End of discovery of accessible devices"
  - "End of topology discovery"
- If autonegotiation is disabled, none of the check boxes can be used.

10.8.6.3 Communication via PROFIBUS

Communication via PROFIBUS

HMI connections via PROFIBUS

If you have inserted an HMI device and a SIMATIC S7 300/400 into the project, you interconnect the two PROFIBUS interfaces in the "Devices & Networks" editor.



HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFIBUS in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFIBUS

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 300/400 via PROFIBUS in the "Devices & Networks" editor.

Requirements

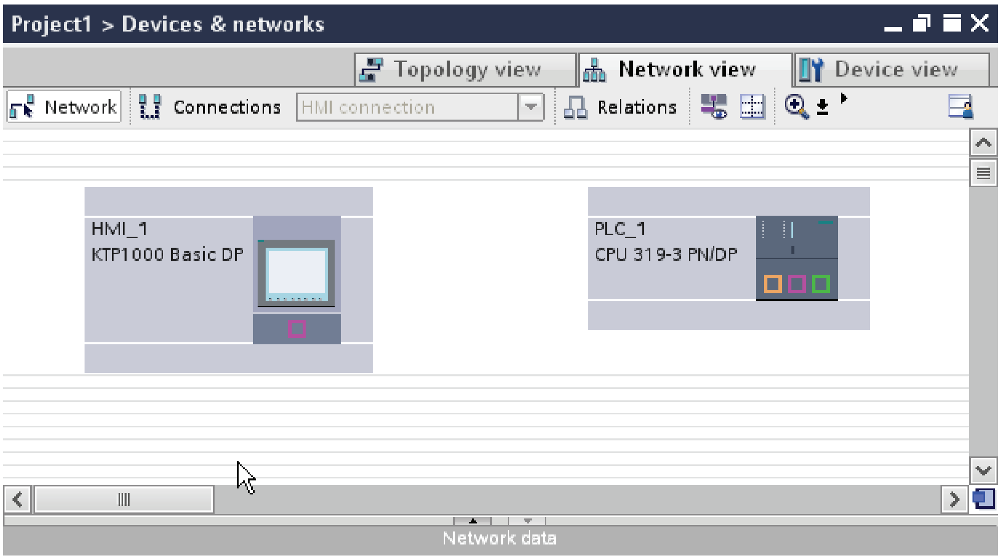
The following communication partners are created in the "Devices & Networks" editor:

- HMI device with MPI/DP interface
- SIMATIC S7 300/400 with PROFIBUS interface

Procedure

1. Double-click the "Devices & Networks" item in the project tree.

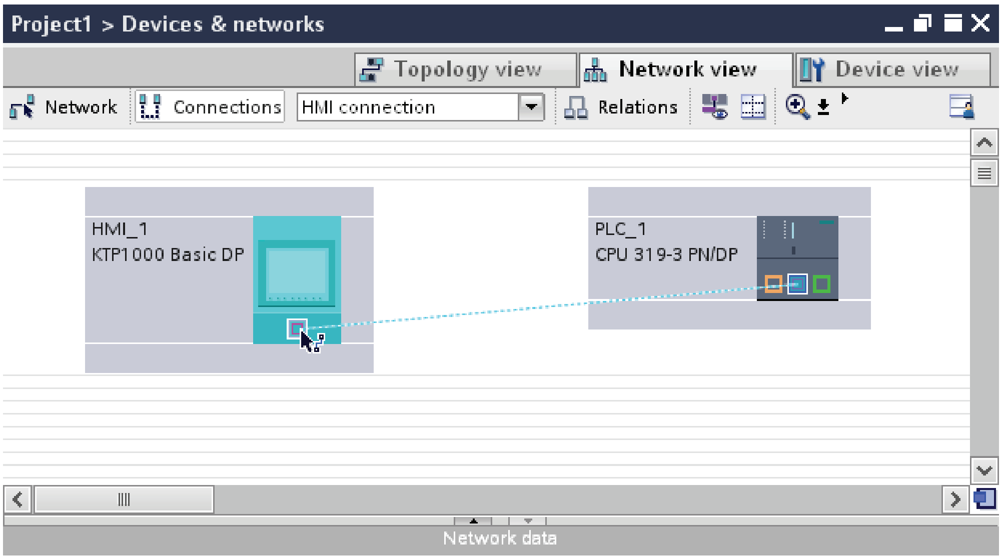
   The available communication partners in the project are displayed graphically in the network view.



2. Click the "Connections" button.

   The devices available for connection are highlighted in color.
3. Click the HMI device interface.
4. Select the "PROFIBUS" interface type in the Inspector window under "Properties > General > HMI MPIDP > Parameters".

5. Click the interface of the PLC and use a drag-and-drop operation to draw a connection to the HMI device.



6. Click the connecting line.
7. Click "Highlight HMI connection" and select the HMI connection.

   The connection is displayed graphically in the Inspector window.
8. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.

   See the chapter "PROFIBUS parameters (Page 2487)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC S7 300/400 via PROFIBUS.

PROFIBUS parameters

PROFIBUS parameters for the HMI connection

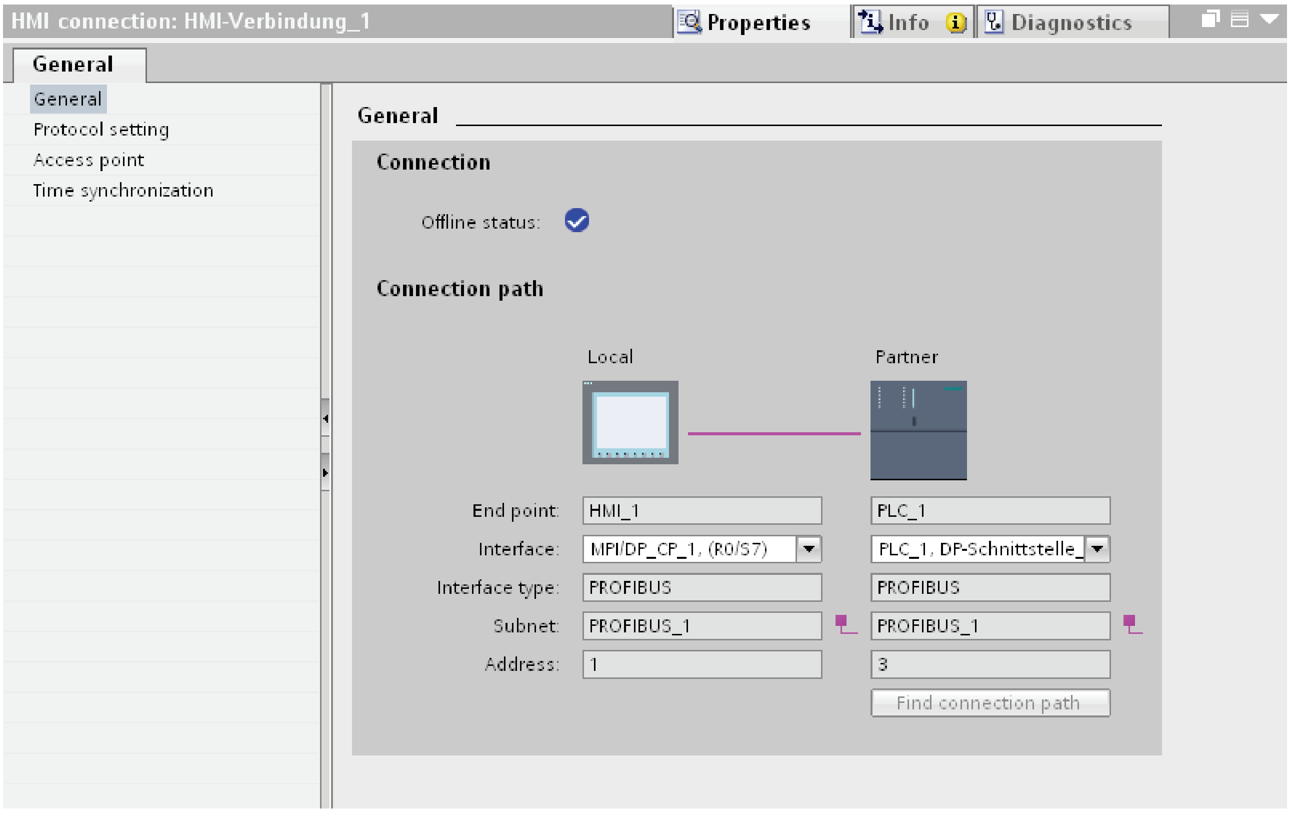
PROFIBUS parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and changing the HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



Offline status: ✓

	Local	Partner
End point:	HMI_1	PLC_1
Interface:	MPI/DP_CP_1, (R0/S7)	PLC_1, DP-Schnittstelle_
Interface type:	PROFIBUS	PROFIBUS
Subnet:	PROFIBUS_1	PROFIBUS_1
Address:	1	3

"Connection"

Displays whether the devices are networked together.

✓ - displayed if the devices are networked together.

! - displayed if the devices are not networked together.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFIBUS parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"

  Displays the device name. This area is not editable.
- "Interface"

  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"

  Displays the selected interface type. This area is not editable.
- "Subnet"

  Displays the selected subnet. This area is not editable.
- "Address"

  Displays the PROFIBUS address of the device. This area is not editable.
- "Find connection path" button

  Enables the subsequent specification of connections.

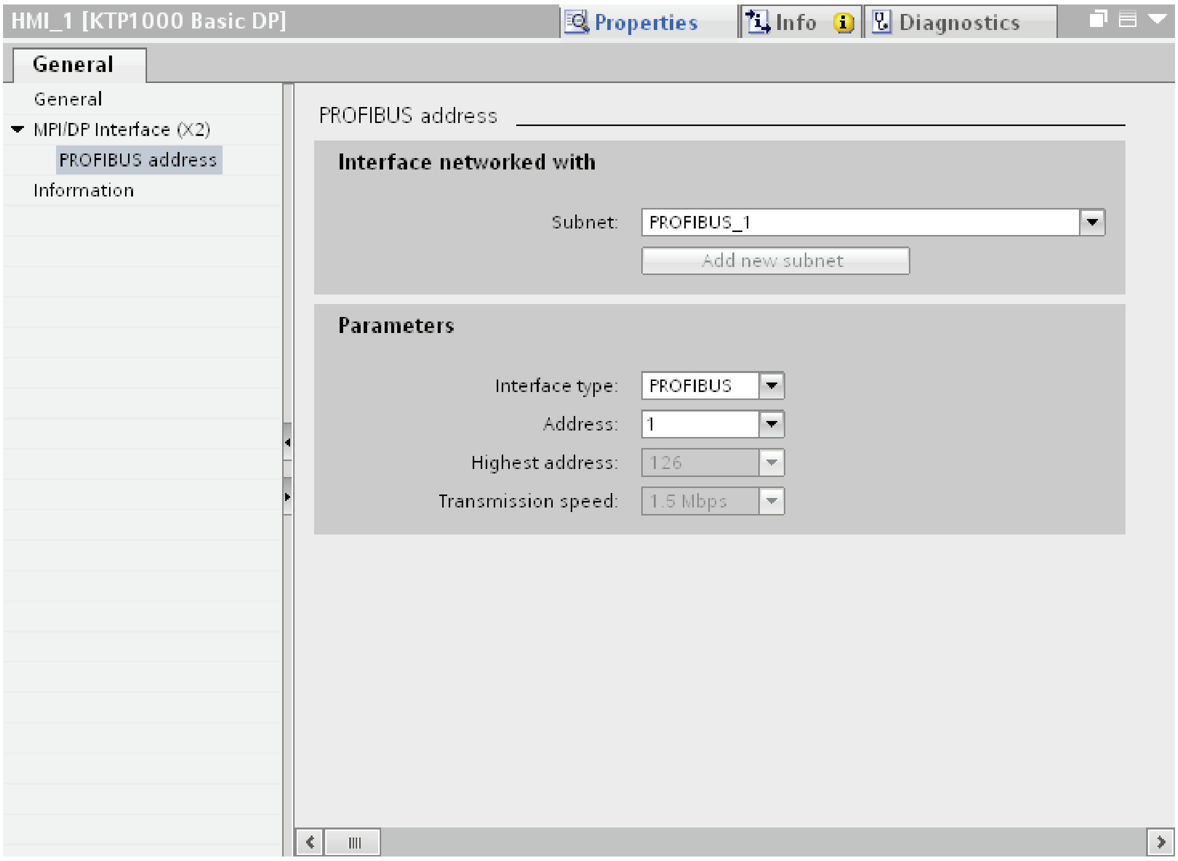
PROFIBUS parameters for the HMI device

PROFIBUS parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFINET parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General > General".



Subnet: PROFIBUS_1

Interface type: PROFIBUS

Address: 1

Highest address: 126

Transmission speed: 1.5 Mbps

"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"

  You assign the interface type in the "Interface type" area. Depending on the HMI device type, you have various interfaces to choose from.

- "Address"

  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.

- "Highest address"

  The "Highest address" area displays the highest address of the PROFIBUS network.

- "Transmission speed"

  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

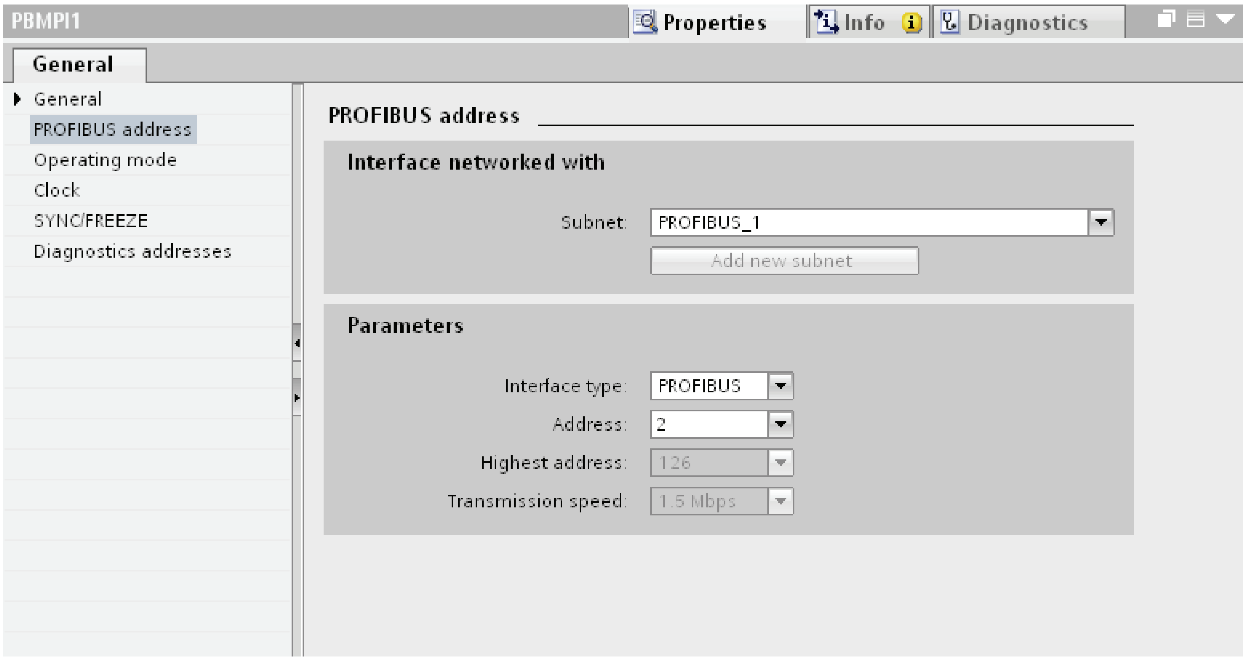
PROFIBUS parameters for the PLC

PROFIBUS parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFIBUS parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



PROFIBUS address

Interface networked with

Subnet: PROFIBUS_1

Parameters

Interface type: PROFIBUS

Address: 2

Highest address: 126

Transmission speed: 1.5 Mbps

"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"

  Depending on the HMI device type, you have various interfaces to choose from.
- "Address"

  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.
- "Highest address"

  The "Highest address" area displays the highest address of the PROFIBUS network.
- "Transmission speed"

  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

Bus profiles with PROFIBUS

Introduction

Depending on the device types connected and protocols used on the PROFIBUS, different profiles are available. The profiles differ in terms of the setting options and calculation of bus parameters. The profiles are explained below.

Devices with different profiles on the same PROFIBUS subnet

The PROFIBUS subnet only functions without problem if the bus parameters of all devices have the same values.

Profiles and transmission rates

Profiles	Supported transmission speeds in Kbits/s
DP	9,6 19,2 45,45 93,75 187,5 500 1500 3000 6000 12000
Standard	9,6 19,2 45,45 93,75 187,5 500 1500 3000 6000 12000
Universal	9,6 19,2 93,75 187,5 500 1500

Meaning of profiles

Profile	Meaning
DP	Select the "DP" bus profile when the only devices connected to the PROFIBUS subnet are those which satisfy the requirements of standard EN 50170 Volume 2/3, Part 8-2 PROFIBUS. The bus parameter setting is optimized on these devices. This includes devices with DP master and DP slave interfaces of the SIMATIC S7 and distributed I/Os of other manufacturers.
Standard	Compared to the "DP" profile, the "Standard" profile also offers scope for devices of another project or devices which have not been configured here to be taken into account when calculating the bus parameters. The bus parameters are then calculated following a simple, non-optimized algorithm.
Universal	Select the "Universal" bus profile when individual devices on the PROFIBUS subnet use the PROFIBUS-FMS service. This includes the following devices for example: • CP 343-5 • PROFIBUS-FMS devices of other manufacturers As with the "Standard" profile, this profile allows you to take other devices into account when calculating the bus parameters.

10.8.6.4 Communication via MPI

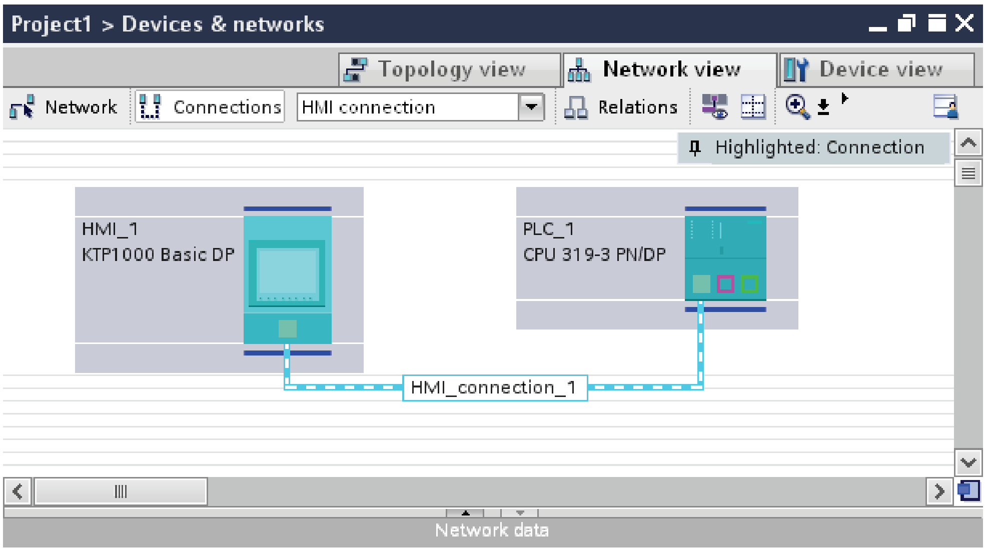
Communication via MPI

HMI connections via MPI

If you have inserted an HMI device and a SIMATIC S7 300/400 into the project, you interconnect the two MPI interfaces in the "Devices & Networks" editor.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFIBUS in the "Devices & Networks" editor.



Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device in the "Connections" editor of the HMI device.

Configuring an HMI connection via MPI

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 300/400 via MPI in the "Devices & Networks" editor.

Requirements

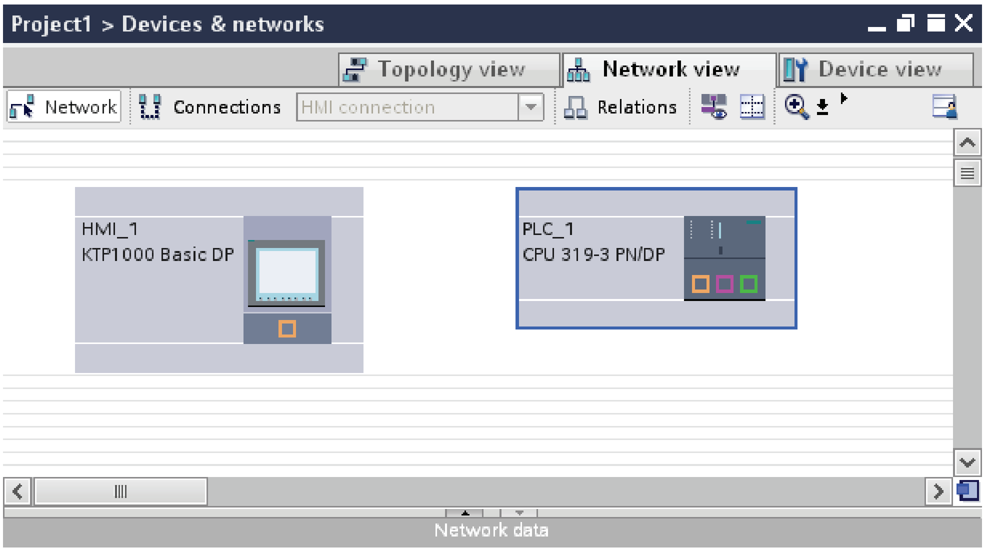
The following communication partners are created in the "Devices & Networks" editor:

- HMI device with MPI/DP interface
- SIMATIC S7 300/400 with MPI/DP interface

Procedure

1. Double-click the "Devices & Networks" item in the project tree.

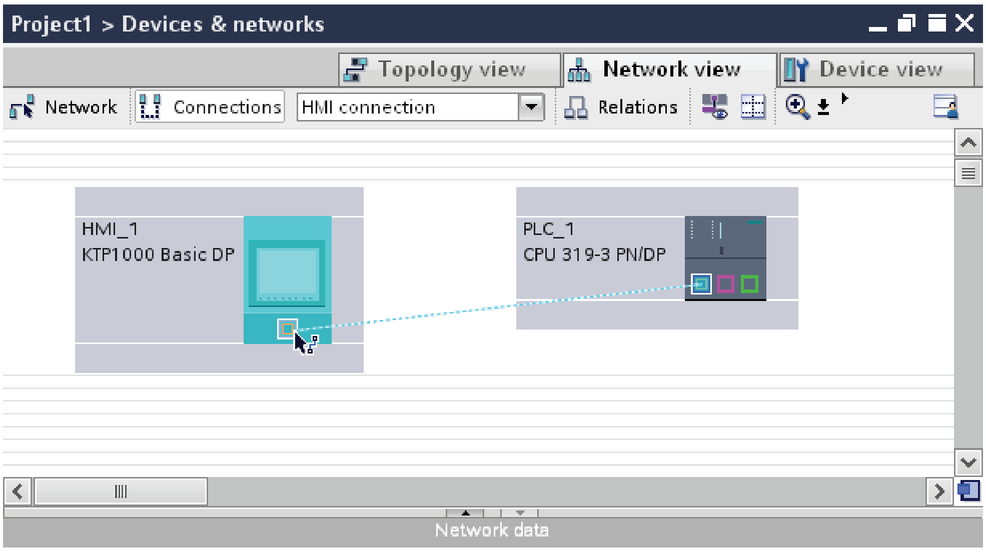
   The available communication partners in the project are displayed graphically in the network view.



2. Click the "Connections" button.

   The devices available for connection are highlighted in color.

3. Click the interface of the PLC and use a drag-and-drop operation to draw a connection to the HMI device.



4. Click the connecting line.

   The connection is displayed graphically in the Inspector window.

5. Click "Highlight HMI connection" and select the HMI connection.
6. Click the communication partners in the "Network view" and change the MPI parameters in the Inspector window according to the requirements of your project.

   See the chapter "MPI parameters (Page 2495)" for additional details.

   Note

   The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. Use the table to monitor the connection parameters and change the connection partner. You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC S7 300/400 via MPI.

MPI parameters

MPI parameters for the HMI connection

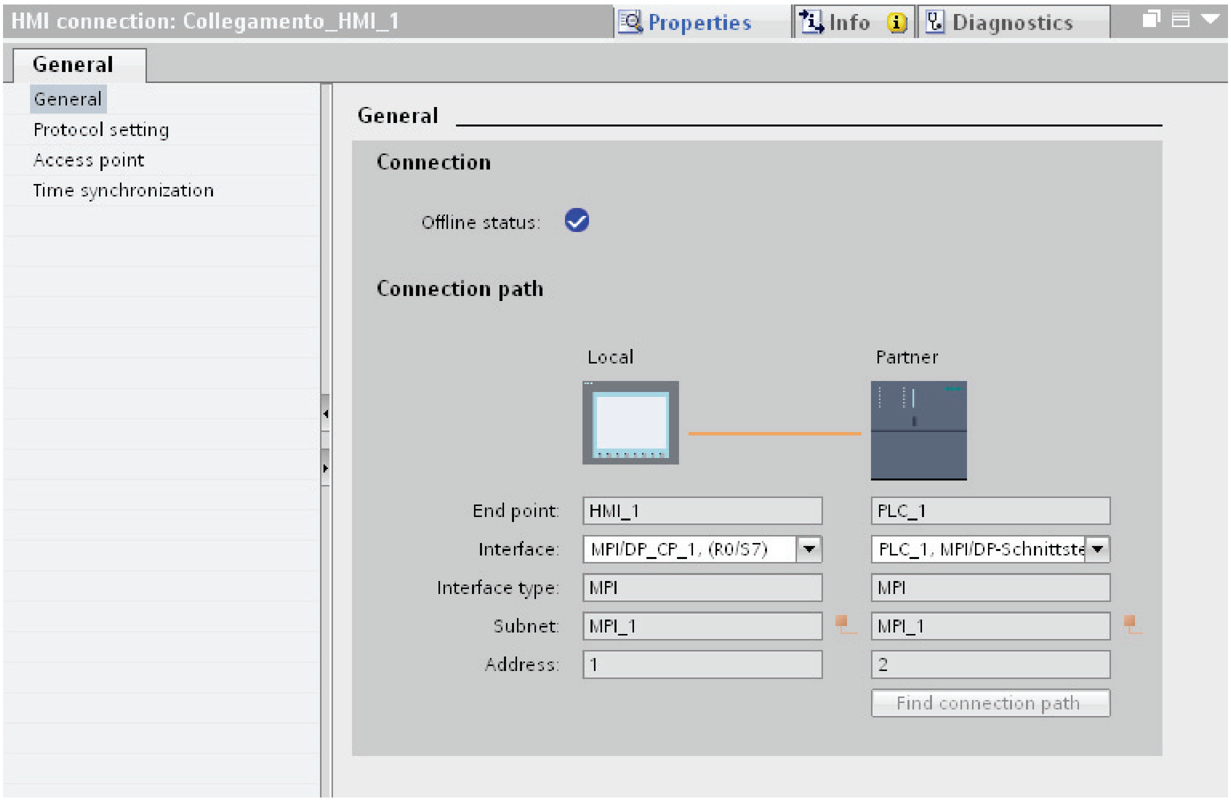
MPI parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and changing the HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



Offline status: ✓

	Local	Partner
End point:	HMI_1	PLC_1
Interface:	MPI/DP_CP_1, (R0/S7)	PLC_1, MPI/DP-Schnittste
Interface type:	MPI	MPI
Subnet:	MPI_1	MPI_1
Address:	1	2

"Connection"

Displays whether the devices are networked together.

✓ - displayed if the devices are networked together.

❗ - displayed if the devices are not networked together.

"Connection path"

The communication partners of the selected HMI connection and the associated MPI parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"

  Displays the name of the device. This area is not editable.
- "Interface"

  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"

  Displays the selected interface type. This area is not editable.
- "Subnet"

  Displays the selected subnet. This area is not editable.
- "Address"

  Displays the MPI address of the device. This area is not editable.
- "Find connection path" button

  Enables the subsequent specification of connections.

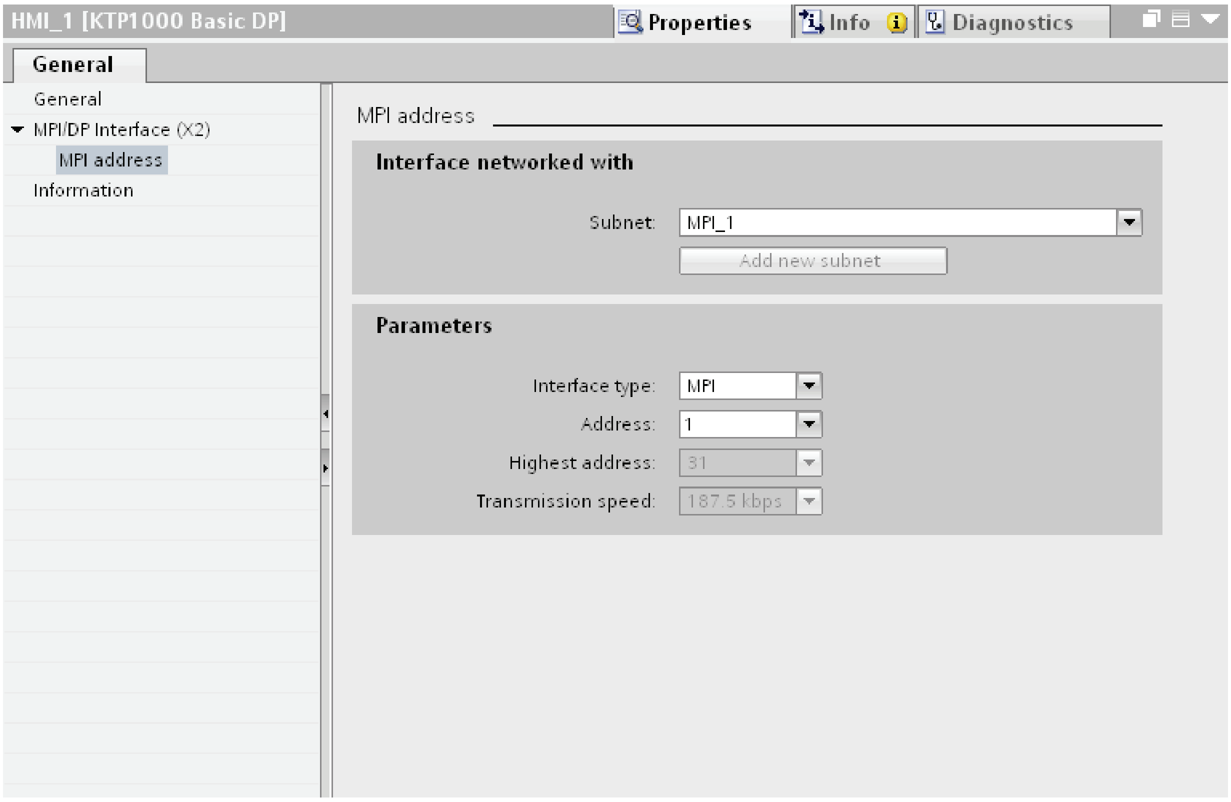
MPI parameters for the HMI device

MPI parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing MPI parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General > General".



Subnet: MPI_1

Interface type: MPI

Address: 1

Highest address: 31

Transmission speed: 187.5 kbps

"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"

  You assign the interface type in the "Interface type" area. Depending on the HMI device type, you have various interfaces to choose from.
- "Address"

  You assign the MPI address of the HMI device in the "Address" area. The MPI address must be unique throughout the MPI network.
- "Highest address"

  The "Highest address" area displays the highest address of the MPI network.
- "Transmission speed"

  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

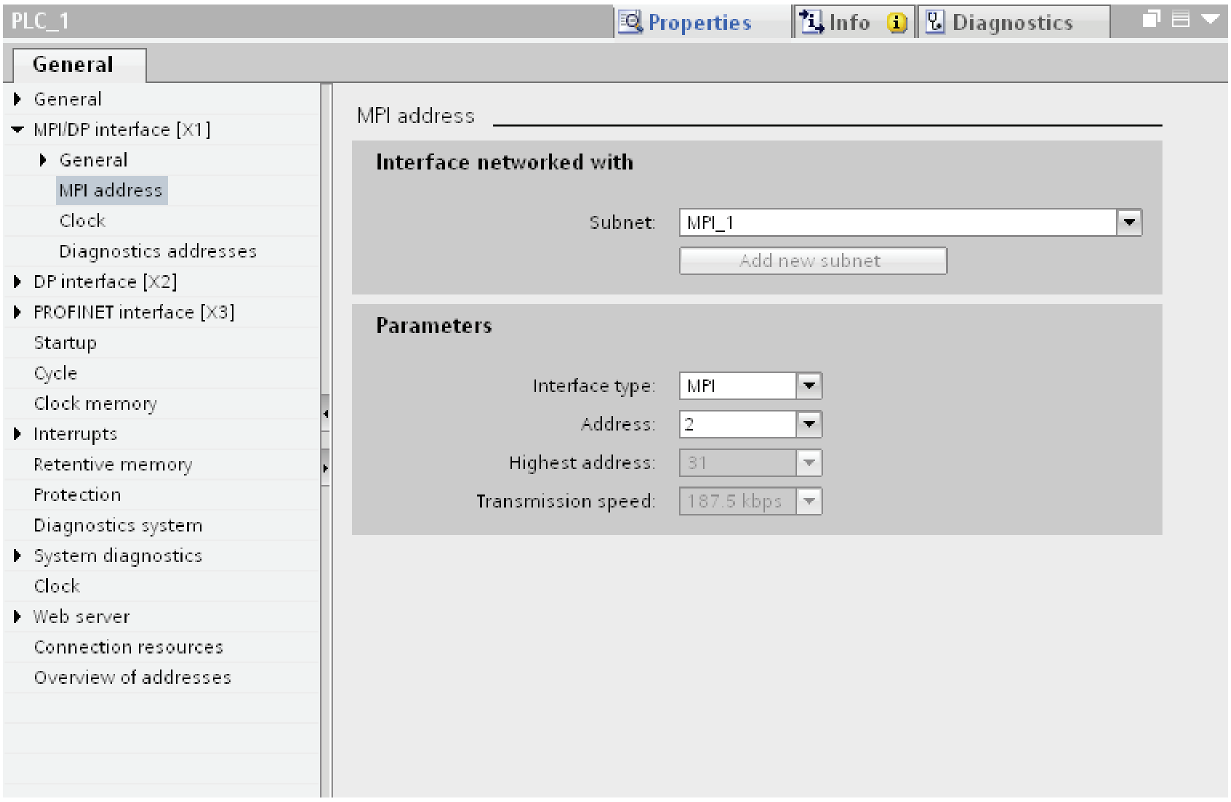
MPI parameters for the PLC

MPI parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFIBUS parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



MPI address

Interface networked with

Subnet: MPI_1

Parameters

Interface type: MPI

Address: 2

Highest address: 31

Transmission speed: 187.5 kbps

"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"

  Depending on the HMI device type, you have various interfaces to choose from.
- "Address"

  You assign the MPI address of the HMI device in the "Address" area. The MPI address must be unique throughout the MPI network.
- "Highest address"

  The "Highest address" area displays the highest address of the MPI network.
- "Transmission speed"

  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

Addressing of the PLC via MPI

Introduction

Each communication partner must be assigned an MPI network address.

Each S7 module which supports communication functions and is operated the SIMATIC S7-300/400 PLC is assigned a unique MPI address. Only one CPU may be used per rack.

Note

HMI devices cannot be operated with incorrect addressing

Always avoid redundant addressing on the MPI bus.

MPI address of the communication partner of a SIMATIC S7-300

When assigning addresses, you have to distinguish between communication partners with and without separate MPI address.

- If the communications partner has its own MPI address, you only need to define the MPI address.
- If the communication partners do not have a separate MPI address, specify the MPI address of the communications partner used for the connection. In addition, define the slot and rack of a communication partner without its own MPI address.

MPI address of the communication partner of a SIMATIC S7-400

Only S7 modules with an MPI connector are assigned an MPI address. Modules without an MPI connector are addressed indirectly:

- MPI address of the module to which the HMI is connected.
- The slot and the rack of the module with which the HMI device communicates.

10.8.6.5 Data exchange

Data exchange using area pointers

General information on area pointers

Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

Configuration of area pointers

Before you use an area pointer, you enable it under "Connections > Area pointer". You then assign the area pointer parameters.

You can find more detailed information on configuring area pointers in:

Configuring area pointers (Page 2410)

"Screen number" area pointer

Function

The HMI device saves information about the screen called on the HMI device to the "Screen number" area pointer.

This allows the transfer of the current screen contents from the HMI device to the PLC. The PLC can trigger specific reactions such as the call of a different screen.

Use

Configure and enable the area pointer in "Communication > Connections" before you put it into use. You can create only one instance of the "Screen number" area pointer and only on one PLC.

The screen number is always transferred to the PLC when a new screen is activated or when the focus within a screen changes from one screen object to another.

Structure

The area pointer is a data area in the memory of the PLC with a fixed length of 5 words.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1st word	Current screen type															
2nd word	Current screen number															
3rd word	Reserved															
4th word	Current field number															
5th word	Reserved															

- Current screen type

  "1" for root screen or
  "4" for permanent window
- Current screen number

  1 to 32767
- Current field number

  1 to 32767

"Date/time" area pointer

Function

This area pointer is used to transfer the date and time from the HMI device to the PLC.

The PLC writes control job "41" or "40" to the job mailbox.

When it evaluates the control job, the HMI device writes its current date and the time in the data area configured in the "Date/time" area pointer. All definitions are coded in BCD format.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

The date/time data area has the following structure:

Data word	Most significant byte								Least significant byte								
	7							0	7							0	
n+0	Reserved								Hour (0 to 23)								Time
n+1	Minute (0 to 59)								Second (0 to 59)								
n+2	Reserved								Reserved								
n+3	Reserved								Weekday (1 to 7, 1=Sunday)								Date
n+4	Day (1 to 31)								Month (1 to 12)								
n+5	Year (80 to 99/0 to 29)								Reserved								

Note

Note that when you enter the year, values 80 to 99 result in years 1980 through 1999 and the values 0 to 29result in the years 2000 through 2029.

"Date/time PLC" area pointer

Function

This area pointer is used to transfer the date and time from the PLC to the HMI device. Use this area pointer if the PLC is the time master.

The PLC loads the data area of the area pointer. All definitions are coded in BCD format.

The HMI device reads the data cyclically within the configured acquisition cycle and synchronizes itself.

Note

Set an acquisition cycle of sufficient length for the date/time area pointer in order to avoid any negative impact on HMI device performance.
Recommended: Acquisition cycle of 1 minute, if the process allows this.

The date/time data area has the following structure:

DATE_AND_TIME format (in BCD code)

Data word	Most significant byte	Least significant byte	
	7 0	7 0	
n+0	Year (80 to 99/0 to 29)	Month (1 to 12)	
n+1	Day (1 to 31)	Hour (0-23)	
n+2	Minute (0-59)	Second (0 to 59)	
n+3	Reserved	Reserved	Weekday (1 to 7, 1=Sunday)
n+4 [1)]	Reserved	Reserved	
n+5 [1)]	Reserved	Reserved	

1) The two data words must exist in the data area to ensure that the data format corresponds to WinCC flexible and to avoid false information being read.

Note

Note that when you enter the year, values 80 to 99 result in years 1980 through 1999 and the values 0 to 29 result in the years 2000 through 2029.

"Coordination" area pointer

Function

The "Coordination" area pointer is used to implement the following functions:

- Detecting the startup of the HMI device in the control program
- Detecting the current operating mode of the HMI device in the control program
- Detecting whether the HMI device is ready to communicate in the control program

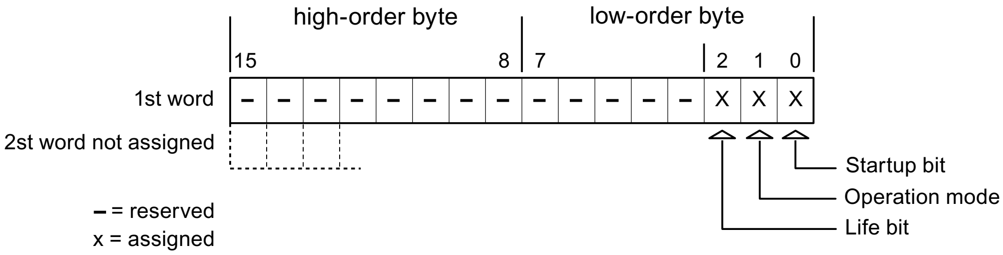
The "Coordination" area pointer has a length of two words.

Use

Note

The HMI device always writes the entire coordination area when updating the area pointer. The control program can for this reason not make changes to the coordination area.

Assignment of bits in the "Coordination" area pointer



Startup bit

The startup bit is set briefly to "0" by the HMI device during startup. It sets the bit permanently to "1" when startup is completed.

Operating mode

The operating mode bit is set to 1 as soon as the user switches the HMI device offline. The status of the operating mode bit is "0" during normal operation of the HMI device. You can determine the current operating mode of the HMI device by reading this bit in the control program.

Life bit

The HMI device inverts the life bit at intervals of approximately one second. You can check whether or not there is still a connection to the HMI device by querying this bit in the control program.

"Project ID" area pointer

Function

When Runtime starts it can check to see if the HMI device is connected to the correct PLC. This check is important when operating with several HMI devices.

The HMI device compares a value stored on the PLC with the value specified in the configuration data. This ensures the compatibility of the configuration data and the PLC program. If there is no concordance, a system alarm is given on the HMI device and Runtime is stopped.

Use

To use this area pointer, set up the following during the configuration:

- Define the version of configuration. Possible values between 1 and 255.

  You enter the version in the editor "Runtime settings > General" in the "Identification" area.
- Data address of the value for the version that is stored in the PLC:

  Enter the data address in the "Communication > Connections" editor in "Address".

Connection failure

A connection failure to a device on which the "project ID" area pointer is configured results in all the other connections in the project being switched to "offline".

This behavior has the following prerequisites:

- You have configured several connections in a project.
- You are using the "project ID" area pointer in at least one connection.

The following causes which may set connections to "offline":

- The PLC is not available.
- The connection has been switched offline in the engineering system.

"Job mailbox" area pointer

Function

The PLC can use the job mailbox to transfer jobs to the HMI device to trigger corresponding actions on the HMI device. These functions include:

- Display screen
- Set date and time

Data structure

The first word of the job mailbox contains the job number. Depending on the job mailbox, up to three parameters can be transferred.

Word	Most significant byte	Least significant byte
n+0	0	Job number
n+1	Parameter 1	
n+2	Parameter 2	
n+3	Parameter 3	

The HMI device evaluates the job mailbox if the first word of this job is not equal to zero. This means that the parameters must be entered in the job mailbox first, followed by the job number.

Once the HMI device has accepted the job mailbox, the first word is set to 0 again. The execution of the job mailbox is generally not completed at this point in time.

Job mailboxes

All job mailboxes and their parameters are listed below. The "No." column contains the job number of the job mailbox. Job mailboxes can only be triggered by the PLC when the HMI device is online.

No.	Function	
14	Setting the time (BCD coded)	
	Parameter 1	Left byte: - Right byte: hours (0-23)
	Parameter 2	Left byte: minutes (0-59) Right byte: seconds (0-59)
	Parameter 3	-
15	Setting the date (BCD code) [3) 4)]	
	Parameter 1	Left byte: - Right byte: weekday (1-7: Sunday-Saturday)
	Parameter 2	Left byte: day (1-31) Right byte: month (1-12)

No.	Function	
14	**Setting the time (BCD coded)**	
	Parameter 3	Left byte: year
23	**User logon**	
	Logs the user on with the name "PLC user" at the HMI device with the group number transferred in parameter 1. The logon is possible only when the transferred group number exists in the project.	
	Parameter 1	Group number 1 to 255
	Parameter 2, 3	-
24	**User logoff**	
	Logs off the current user. (The function corresponds to the "logoff" system function)	
	Parameter 1, 2, 3	-
40	**Transfer date/time to PLC**	
	(in the S7 format DATE_AND_TIME) An interval of at least 5 seconds must be maintained between two successive jobs in order to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
41	**Transfer date/time to PLC**	
	(In OP/MP format) An interval of at least 5 seconds must be maintained between two successive jobs in order to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
46	**Update tags**	
	Causes the HMI device to read the current value of the tags•from the PLC whose update ID matches the value transferred in parameter 1. (Function corresponds to the "UpdateTag" system function.)	
	Parameter 1	1 - 100
49	**Delete alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Warnings" class from the alarm buffer.	
	Parameter 1, 2, 3	-
50	**Delete alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Errors" class from the alarm buffer.	
	Parameter 1, 2, 3	-
51	**Screen selection** 2)	
	Parameter 1	Screen number
	Parameter 2	-
	Parameter 3	Field number
69	**Reading data record from PLC** 1)	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	0: Do not overwrite existing data record 1: Overwrite existing data record
70	**Writing data record from PLC** 1)	

No.	Function	
14	Setting the time (BCD coded)	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	-

1)	Only for devices supporting recipes.
2)	OP 73, OP 77A and TP 177A HMI devices also execute the "Screen selection" job mailbox if the on-screen keyboard is active.
3)	The weekday is ignored on HMI device KTP 600 BASIC PN.
4)	The weekday is ignored when you configure the "Date/Time PLC" area pointer.

"Data record" area pointer

"Data record" area pointer

Function

When data records are transferred between the HMI device and PLC, both partners access common communications areas on the PLC.

Data transfer types

There are two ways of transferring data records between the HMI device and PLC:

- Transfer without synchronization
- Transfer with synchronization via the data mailbox

Data records are always transferred directly, which means that the tag values are read straight from an address or written straight to an address configured for this tag without being redirected via an interim memory.

Initiating the transfer of data records

There are three ways of triggering the transfer:

- Operator input in the recipe view
- Job mailboxes

  The transfer of data records can also be triggered by the PLC.
- Triggering by configured functions

If the transfer of data records is triggered by a configured function or by a job mailbox, the recipe view on the HMI device remains operable. The data records are transferred in the background.

Simultaneous processing of several transfer requests is, however, not possible. In this case, the HMI device rejects the other transfer requests with a system alarm.

Transfer without synchronization

If you select asynchronous transfer of data records between the HMI device and PLC, there is no coordination over the common data areas. It is therefore unnecessary to set up a data area during configuration.

Asynchronous data record transfer can be a useful alternative, for example when:

- The system is capable of excluding the risk of uncontrolled overwriting of data by the communication peer.
- The PLC does not require information about the recipe number and data record number.
- The transfer of data records is triggered by the operator of the HMI device.

Reading values

When a read job is triggered, the values are read from the PLC addresses and transferred to the HMI device.

- Triggering by the operator in the recipe view:

  The values are downloaded to the HMI device. You can then, for example, process, edit, or save these values in the HMI device.
- Triggering by a function or job mailbox:

  The values are saved immediately to the data volume.

Writing values

When a write job is triggered, the values are written to the PLC addresses.

- Triggering by the operator in the recipe view:

  The current values are written to the PLC.
- Triggering by a function or job mailbox:

  The current values are written to the PLC from the data medium.

Transfer with synchronization

If you select synchronous transfer, both communication partners set status bits in the common data area. You can use this mechanism to prevent uncontrolled overwriting of data in either direction in your control program.

Application

Synchronous data record transfer can be a useful solution, for example, when:

- The PLC is the "active partner" in the transfer of data records.
- The PLC evaluates the information about the recipe number and data record number.
- The transfer of data records is triggered by means of a Job mailbox.

Requirements

In order to synchronize transfer of data records between the HMI device and the PLC, the following requirements must be met during configuration:

- An area pointer has been set up: "Communication > Connections" editor in "Area pointer".
- The PLC with which the HMI device synchronizes transfer of data records is specified in the recipe:

  "Recipes" editor in the inspector window the option "Coordinated transfer of data records" under "General > Synchronization > Settings"

Structure of the data area

The data area has a fixed length of 5 words. Structure of the data area:

	15		0
1. Word	Current recipe number (1 - 999)		
2. Word	Current data record number (0 - 65535)		
3. Word	Reserved		
4. Word	Status (0, 2, 4, 12)		
5. Word	Reserved		

- Status

  The status word (word 4) can adopt the following values:

Value		Meaning
Decimal	Binary	
0	0000 0000	Transfer permitted, data mailbox free
2	0000 0010	Transferring
4	0000 0100	Transfer completed without error
12	0000 1100	Transfer completed with error

Sequence of a transfer started by the operator in the recipe view

Reading from the PLC started by the operator in the recipe view

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe number to be read and the status "Transferring" in the data record and sets the data record number to 0.	Abort with system event.

Step	Action	
3	The HMI device reads the values from the PLC and displays them in the recipe view. If the recipes have synchronized tags, the values from the PLC are also written to the tags.	
4	The HMI device sets the status "Transfer completed."	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC started by the operator in the recipe view

Step	Action	
	Check: Status word = 0?	
1	Yes	No
	The HMI device enters the recipe and data record number to be written and the status "Transferring" in the data record.	Abort with system event.
2	The HMI device writes the current values to the PLC. If the recipes have synchronized tags, the changed values are synchronized between the recipe view and tags and then written to the PLC.	
3	The HMI device sets the status "Transfer completed."	
4	If required, the control program can now evaluate the transferred data.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Sequence of the transfer triggered by a job mailbox

The transfer of data records between the HMI device and the PLC can be initiated by either one of these stations.

The two job mailboxs No. 69 and No. 70 are available for this type of transfer.

No. 69: Read data record from PLC ("PLC → DAT")

Job mailbox no. 69 transfers data records from the PLC to the HMI device. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	69
Word 2	Recipe number (1-999)	
Word 3	Data record number (1 to 65535)	
Word 4	Do not overwrite existing data record: 0 Overwrite existing data record: 1	

No. 70: Write data record to PLC ("DAT → PLC")

Job mailbox No. 70 transfers data records from the HMI device to the PLC. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	70
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	—	

Sequence when reading from the PLC with job mailbox "PLC → DAT" (no. 69)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data record.	Abort without return message.
3	The HMI device reads the values from the PLC and saves these to the data record defined in the job mailbox.	
4	• If "Overwrite" was selected in the job, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "Do not overwrite" was selected in the job, and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Sequence of writing to the PLC with job mailbox "DAT → PLC" (no. 70)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device fetches the values of the data record specified in the function from the data medium and writes the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Sequence of the transfer when triggered by a configured function

Reading from the PLC using a configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.
3	The HMI device reads the values from the PLC and stores them in the data record specified in the function.	
4	• If "Yes" was selected for the "Overwrite" function, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "No" was selected for the "Overwrite" function and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC by means of configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.
3	The HMI device fetches the values of the data record specified in the function from the data medium and transfers the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Possible causes of error when transferring data records

Possible causes of error

The section below shows possible error causes which lead to the cancellation of data record transfer:

- Tag address not set up on the PLC
- Overwriting data records not possible
- Recipe number does not exist
- Data record number does not exist

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Reaction to an aborted transfer due to errors

If the transfer of data records is aborted due to errors, the HMI device reacts as follows:

- Triggering by the operator in the recipe view

  Information in the status bar of the recipe view and output of system alarms
- Triggered by function

  Output of system alarms
- Triggering by job mailbox

  No return message on the HMI device

You can nonetheless evaluate the status of the transfer by querying the status word in the data record.

Trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out time-triggered for Basic Panels.

For additional information see:

Configuring trend displays for values from the PLC (Page 2155)

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration.

Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Alarms

Configuring alarms

Configure alarms

Several steps are needed to configure alarms, such as operational messages, error alarms, and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with alarms (Page 2167)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a SIMATIC communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
SIMATIC S7 300/400	WORD, INT	BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIME

How the bit positions are counted

For connections with a SIMATIC communication driver, the following counting method applies:

How the bit positions are counted	Byte 0								Byte 1							
	Most significant byte								Least significant byte							
In SIMATIC S7 PLCs	7							0	7							0
In WinCC you configure:	15							8	7							0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

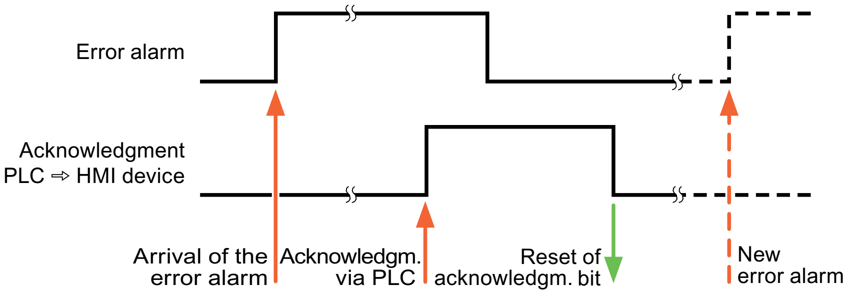
Acknowledgment by the PLC

In "Write acknowledgment tag", you configure the tag or the array tag and the bit number based on which the HMI device can recognize an acknowledgment by the PLC.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.


Error alarm
Acknowledgment PLC ⇨ HMI device
Arrival of the error alarm
Acknowledgm. via PLC
Reset of acknowledgm. bit
New error alarm

Acknowledgment on the HMI device

In "Read acknowledgment tag", you configure the tag or the array tag and the bit number that is written to the PLC after acknowledgment from the HMI device. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

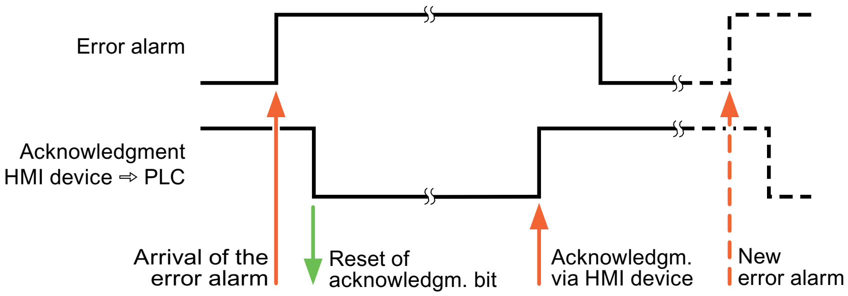
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



10.8.6.6 Performance features of communication

Permitted data types for SIMATIC S7 300/400

Permitted data types for connections with SIMATIC S7 300/400

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Length
BOOL	1-bit
BYTE	1 byte
WORD	2 bytes
DWORD	4 bytes
CHAR	1 byte
INT	2 byte
DINT	4 bytes
REAL	4 bytes
TIME	4 bytes
DATE	2 bytes
TIME_OF_DAY, TOD	4 bytes
S5TIME	2 bytes
COUNTER	2 bytes
TIMER	2 bytes
DATE_AND_TIME	8 bytes
STRING	(2+n) bytes, n = 0 to 254

10.8.6.7 Creating connections in the "Connections" editor

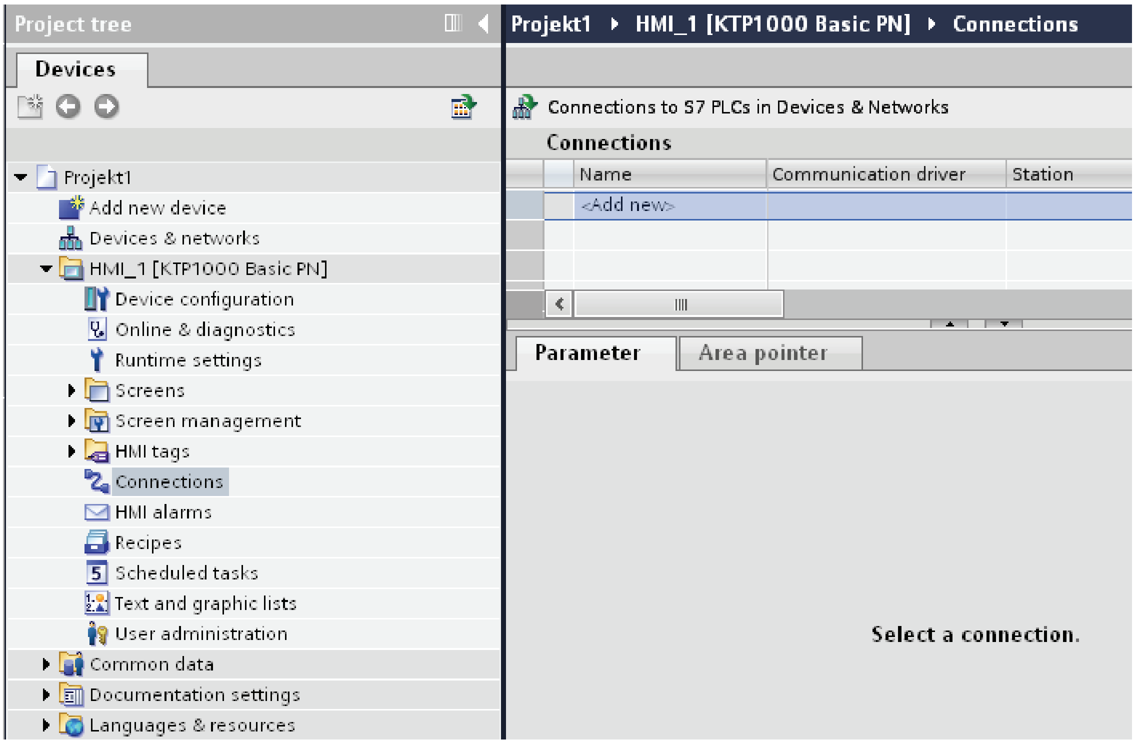
Creating a PROFINET connection

Requirements

- A project is open.
- An HMI device with a PROFINET interface has been created.

Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".

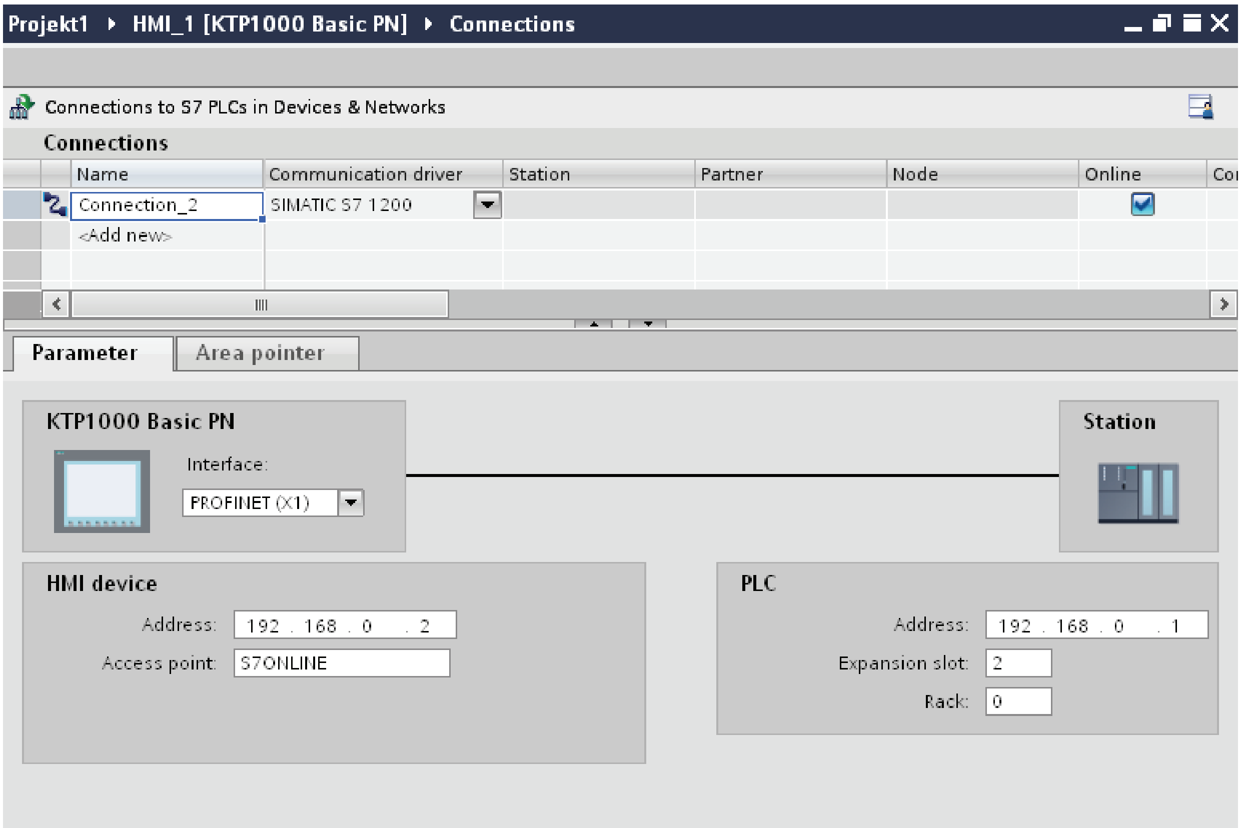


Connections

Name	Communication driver	Station
<Add new>		

Select a connection.

3. Select the driver in the "Communication driver" column.



Projekt1 ▸ HMI_1 [KTP1000 Basic PN] ▸ Connections

Connections to S7 PLCs in Devices & Networks

Connections

Name	Communication driver	Station	Partner	Node	Online	Co
Connection_2	SIMATIC S7 1200				☑	
<Add new>						

Parameter | Area pointer

KTP1000 Basic PN

Interface: PROFINET (X1)

Station

HMI device

Address: 192 . 168 . 0 . 2

Access point: S7ONLINE

PLC

Address: 192 . 168 . 0 . 1

Expansion slot: 2

Rack: 0

4. Click the name of the connection.
5. Select a PROFINET interface of the HMI device in the Inspector window under "Parameters > Interface".
6. Set the IP addresses of the communication partners in the Inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

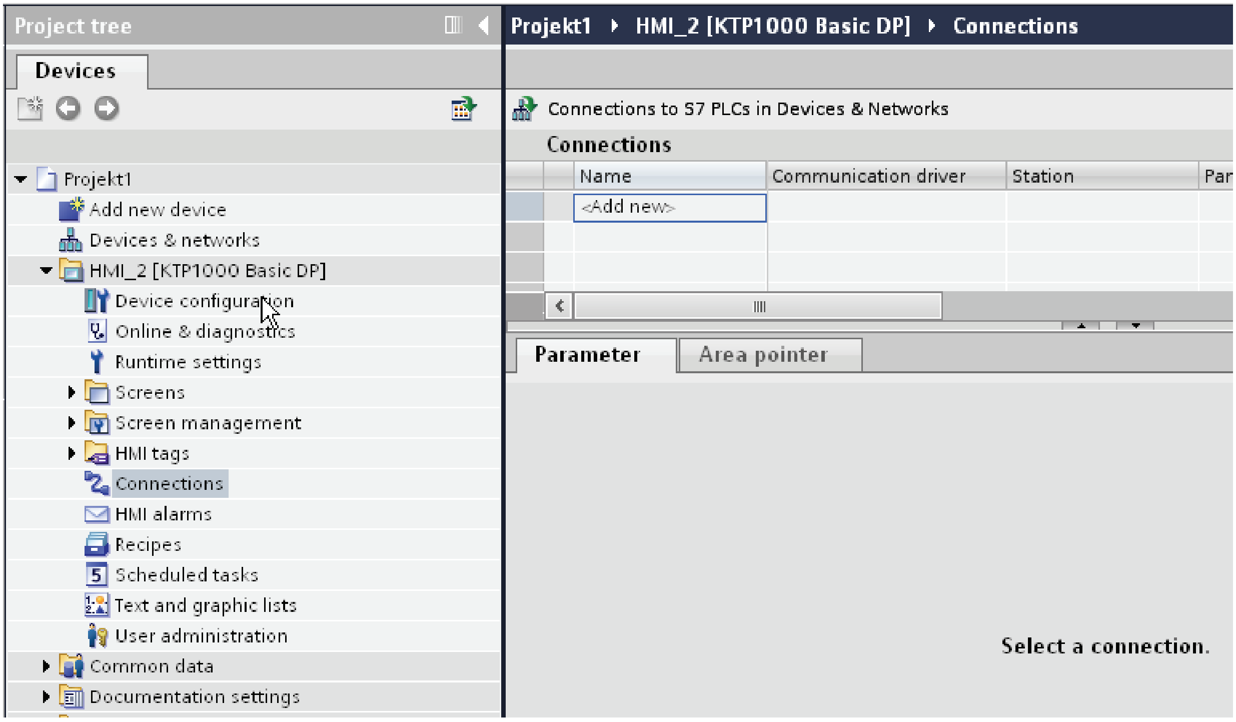
Creating a PROFIBUS connection

Requirements

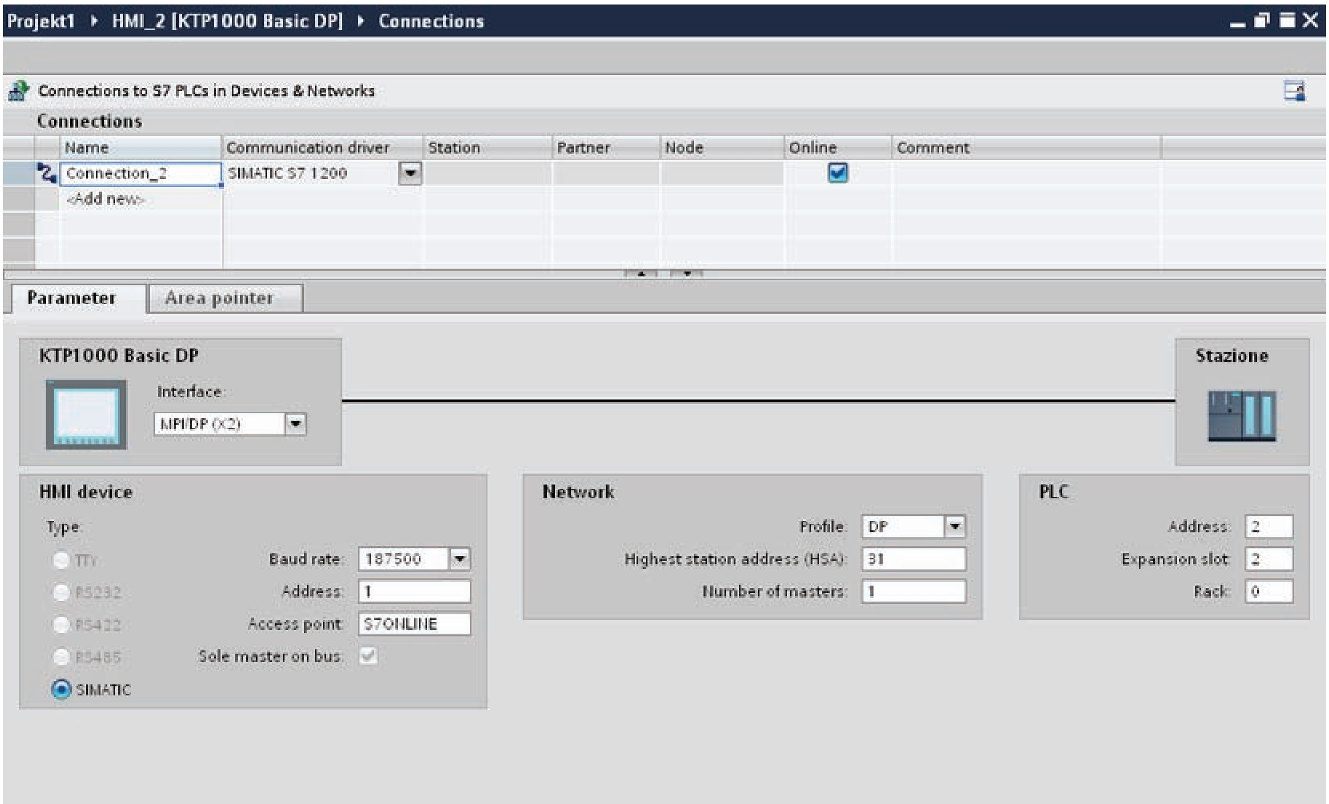
- A project is open.
- An HMI device with a PROFIBUS interface has been created.

Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. Select the driver in the "Communication driver" column.
4. Click the name of the connection.
5. Select the "MPI/DP" interface in the Inspector window under "Parameters > Interface".
6. Select the "DP" profile in the Inspector window under "Parameters > Network".



Projekt1 ▸ HMI_2 [KTP1000 Basic DP] ▸ Connections

Connections to S7 PLCs in Devices & Networks

Connections

Name	Communication driver	Station	Partner	Node	Online	Comment
Connection_2	SIMATIC S7 1200				☑	
<Add new>						

Parameter | Area pointer

KTP1000 Basic DP

Interface: MPI/DP (X2)

Stazione

HMI device

Type:

○ TTY
○ RS232
○ RS422
○ RS485
● SIMATIC

Baud rate: 187500
Address: 1
Access point: S7ONLINE
Sole master on bus: ☑

Network

Profile: DP
Highest station address (HSA): 31
Number of masters: 1

PLC

Address: 2
Expansion slot: 2
Rack: 0

7. Set the addresses of the communication partners in the inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

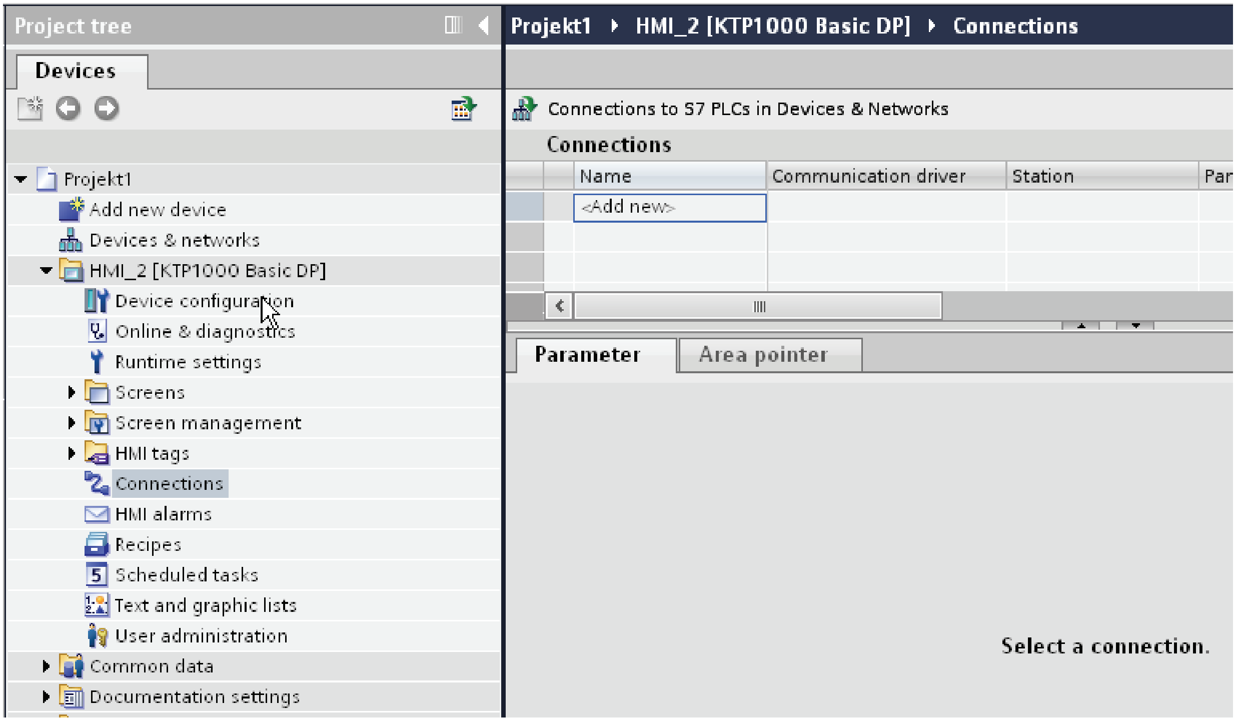
Creating an MPI connection

Requirements

- A project is open.
- An HMI device with an MPI interface has been created.

Procedure

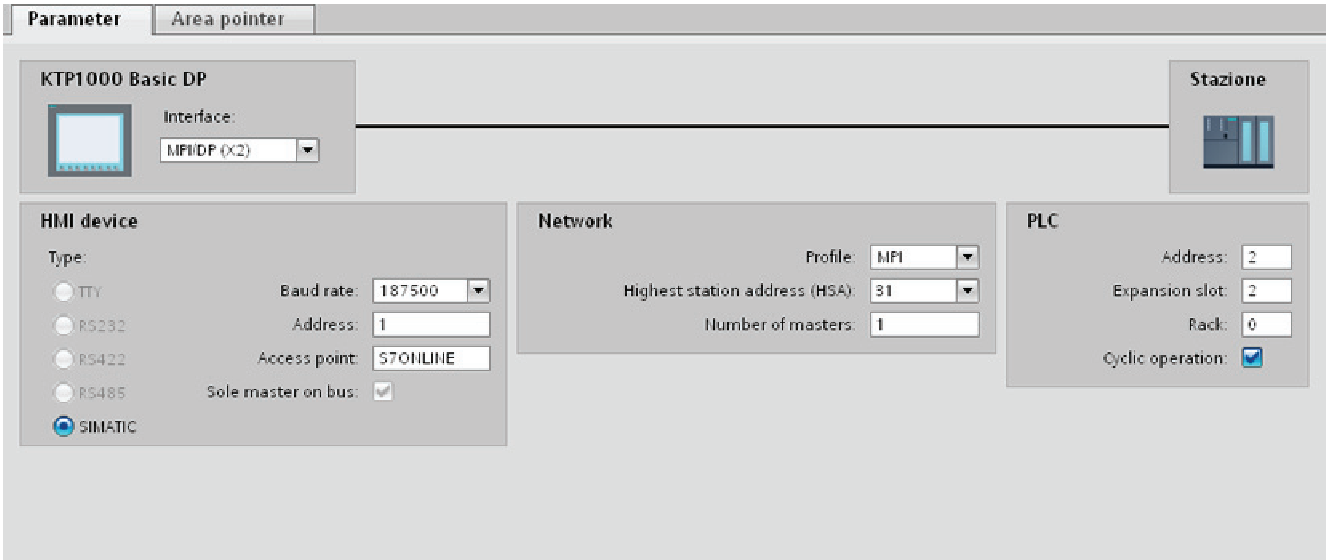
1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



Name	Communication driver	Station	Par
<Add new>			

Select a connection.

3. Select the driver in the "Communication driver" column.
4. Click the name of the connection.
5. Select the "MPI/DP" interface in the Inspector window under "Parameters > Interface".
6. Select the "MPI" profile in the Inspector window under "Parameters > Network".



KTP1000 Basic DP

Interface: MPI/DP (X2)

Stazione

HMI device

Type:

○ TTY
○ RS232
○ RS422
○ RS485
● SIMATIC

Baud rate: 187500
Address: 1
Access point: S7ONLINE
Sole master on bus: ☑

Network

Profile: MPI
Highest station address (HSA): 31
Number of masters: 1

PLC

Address: 2
Expansion slot: 2
Rack: 0
Cyclic operation: ☑

7. Set the addresses of the communication partners in the inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

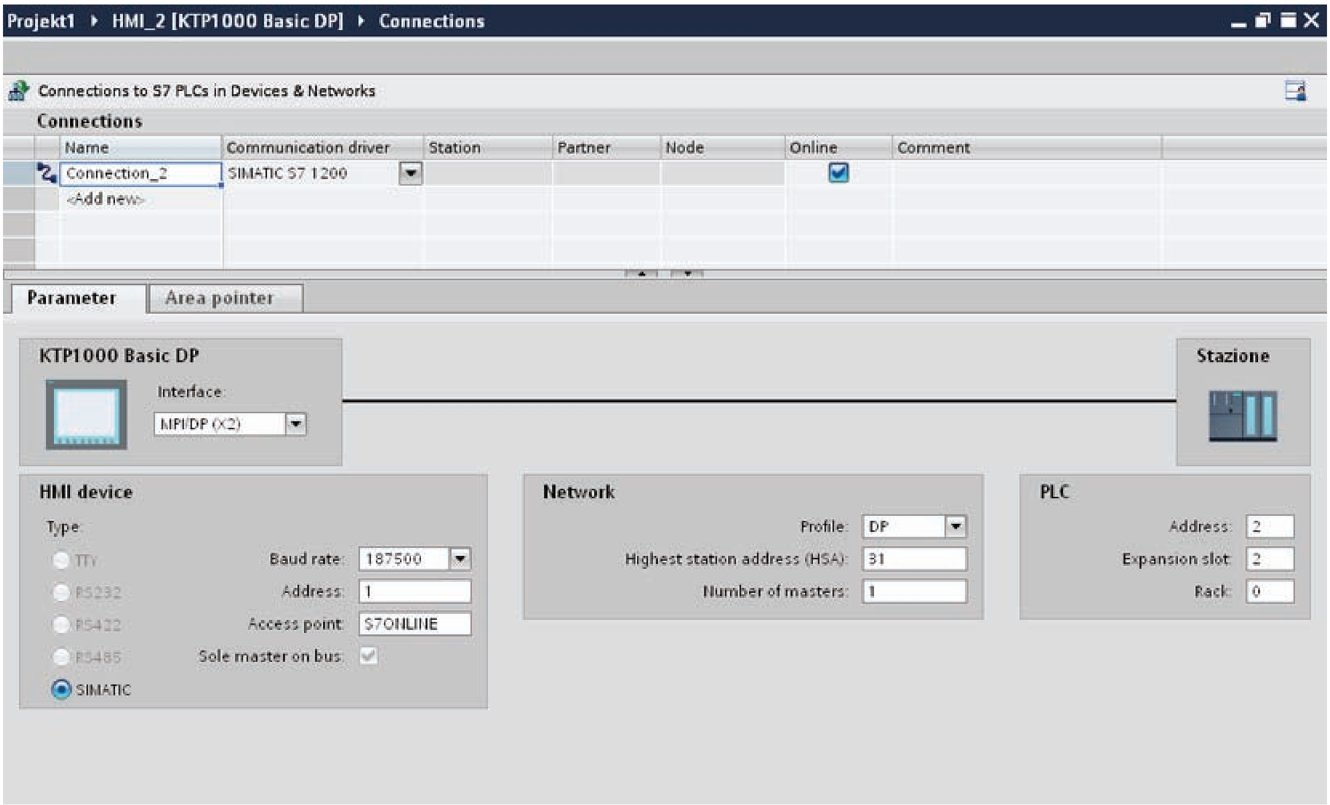
Parameters for the connection

Parameters for the connection (SIMATIC S7 300/400)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device", "Network", and "PLC" areas are available for assigning parameters according to the interface used.



Projekt1 ▸ HMI_2 [KTP1000 Basic DP] ▸ Connections

Connections to S7 PLCs in Devices & Networks

Connections

Name	Communication driver	Station	Partner	Node	Online	Comment
Connection_2	SIMATIC S7 1200				☑	
<Add new>						

Parameter | Area pointer

KTP1000 Basic DP

Interface: MPI/DP (X2)

Stazione

HMI device

Type:

○ TTY
○ RS232
○ RS422
○ RS485
● SIMATIC

Baud rate: 187500
Address: 1
Access point: S7ONLINE
Sole master on bus: ☑

Network

Profile: DP
Highest station address (HSA): 31
Number of masters: 1

PLC

Address: 2
Expansion slot: 2
Rack: 0

Ethernet parameters

Parameters for the HMI device

You set the parameters for the HMI device in the network under "HMI device".. The changes are not transferred automatically to the HMI device. You must change the settings in the Control Panel of the HMI device.

- "Interface"

  If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC.

  Note

  The IP address in the Control Panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

  The IP address already set up in the Control Panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

  The IP address is transferred to the HMI device during project transfer.

To set up the IP address of the HMI device:

- Click the HMI device.
- Open the "Device configuration" editor.
- Click the Ethernet interface.
- Assign the IP address in the inspector window under:

  "General > PROFINET interface > Ethernet addresses"

- "Address"

  You assign the IP address of the HMI device in the "Address" area.

  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.

- "Access point"

  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"

  Under "Address", set the IP address of the S7 module to which the HMI device is connected.

- "Expansion slot"

  Defines the number of the expansion slot of the CPU to be addressed.

- "Rack"

  Defines the rack number of the CPU to be addressed.

- "Cyclic mode"

  Note

  The setting "Cyclic operation" cannot be configured for the SIMATIC S7 1200 PLC.

  When cyclic operation is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This increases system performance.

  Disable cyclic mode if you are operating several HMI devices in parallel.

PROFIBUS parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"

  Specifies the physical connection used.

- "Interface"

  For "Interface", you select the HMI device interface via which the HMI device is connected to the PROFIBUS network.

- "Baud rate"

  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"

  You set the PROFIBUS address of the HMI device under "Address". The PROFIBUS address must be unique in the PROFIBUS network.

- "Sole master on bus"

  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master).

  In S7 200, you must set an HMI device as the master.

- "Access point"

  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the network

Under "Network", you set the parameters for the PROFIBUS network to which the HMI device is linked.

- "Profile"

  For "Profile", you select the network profile that is used in the network. In "Profile", set "DP", "Universal", or "Standard". The setting must be identical throughout the network.

- "Highest address"

  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual PROFIBUS address. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud in the OP 73 or the OP 77A, the highest station address must be less than or equal 63.

- "Number of masters"

  For "Number of masters", set the number of masters in the PROFIBUS network. This information is necessary to correctly calculate the bus parameters.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"

  For "Address", set the PROFIBUS address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.

- "Cyclic operation"

  Note

  The setting "Cyclic operation" cannot be configured for the SIMATIC S7 1200 PLC.

  When cyclic operation is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This increases system performance. Disable cyclic mode if you are operating several HMI devices in parallel. This setting is not required for SIMATIC S7-200 PLCs.

MPI parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"

  Specifies the physical connection used.
- "Interface"

  For "Interface", you select the HMI device interface via which the HMI device is connected to the MPI network.
- "Baud rate"

  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.
- "Address"

  For "Address", you set the MPI address of the HMI device. The MPI address must be unique throughout the MPI network.
- "Sole master on bus"

  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master). If you have connected only slaves to the HMI device, you must therefore disable the "Sole master on bus" safety feature.

  In S7 200, you must set an HMI device as the master.

Parameters for the network

Under "Network", you set the parameters for the MPI network to which the HMI device is linked.

- "Profile"

  For "Profile", you select the network profile that is used in the network. In "Profile", set "MPI". The setting must be identical throughout the network.
- "Highest address"

  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual MPI address. The setting must be identical throughout the network.
- "Number of masters"

  This setting is not required for MPI.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"

  For "Address", set the MPI address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.

- "Cyclic mode"

  When cyclic mode is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This improves system performance. Disable cyclic mode if you are operating several HMI devices in parallel. This setting is not required for SIMATIC S7-200.

Cyclic operation

Handling the "Cyclic operation" selection

If "Cyclic operation" is enabled, the HMI device sends a message frame to the CPU at the beginning of communication indicating that certain tags are required on a recurring basis.

The CPU then always transmits the data at the same cyclic interval. This saves the HMI device from having to output new requests for the data.

If cyclic operation is disabled, the HMI device sends a request whenever information is required.

Additional properties:

- Cyclic operation reduces data transmission load at the HMI device. The PLC resources are used to relieve load on the HMI device.
- The PLC only supports a certain number of cyclic services. The HMI device handles the operation if the PLC cannot provide any further resources for cyclic services.
- The HMI device generates the cycle if the PLC does not support the cyclic mode.
- Screen tags are not integrated in cyclic operation.
- Cyclic mode is only set up at the restart of Runtime.
- The HMI device transfers several jobs to the PLC if cyclic mode is enabled, depending on the PLC.
- The HMI device only transfers one job to the PLC if cyclic mode is disabled.

10.8.7 Communicating with SIMATIC S7 200

10.8.7.1 Communication with SIMATIC S7 200

Introduction

This section describes the communication between an HMI device and the SIMATIC S7 200 PLC.

You can configure the following communication channels for the SIMATIC S7 200 PLC:

- PROFINET and Ethernet
- PROFIBUS
- MPI
- PPI

HMI connection for communication

You configure connections between the HMI device and a SIMATIC S7 200 in the "Connections" editor of the HMI device.

10.8.7.2 Creating a connection to SIMATIC S7 200

Introduction

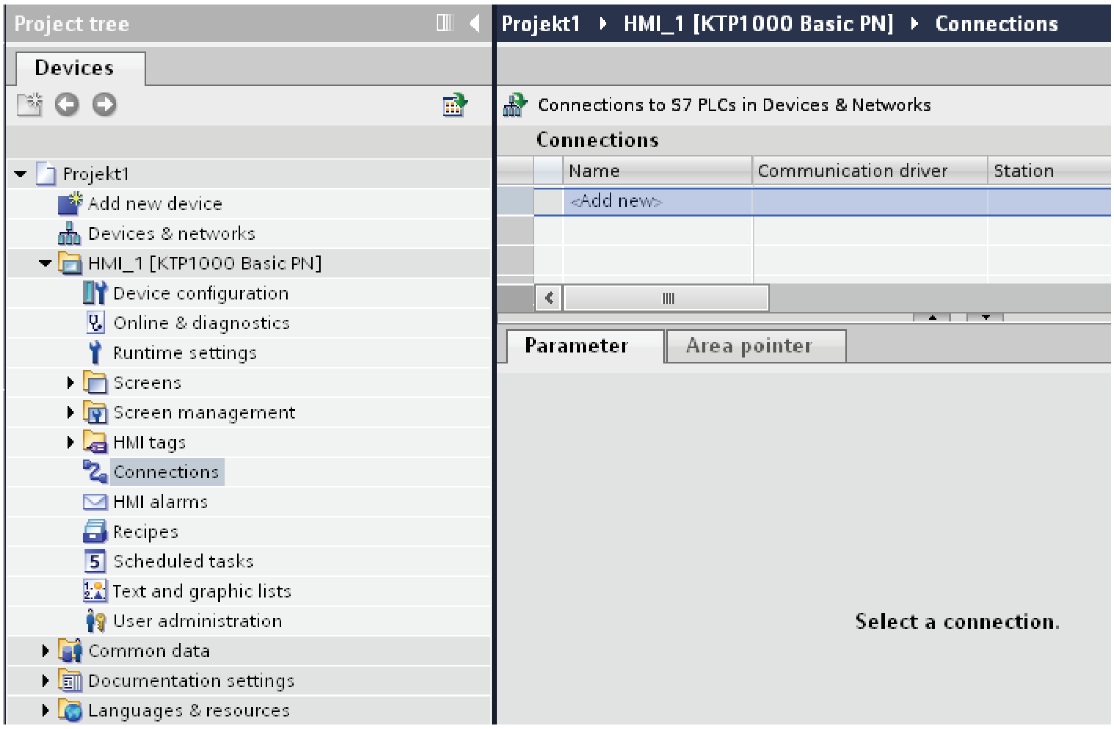
You configure a connection to the SIMATIC S7 200 PLC in the "Connections" editor of the HMI device. The interfaces are named differently depending on the HMI device.

Requirements

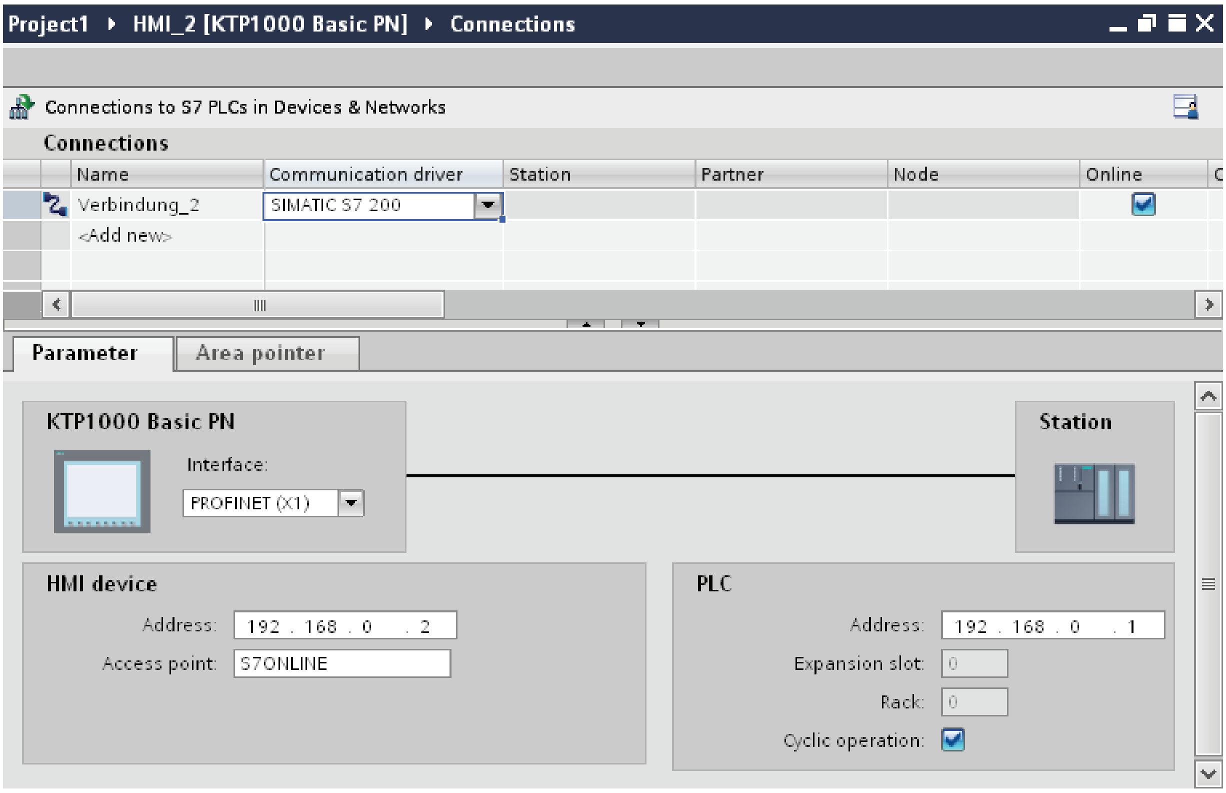
- A project is open.
- An HMI device has been created.

Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.
3. Double-click "<Add>" in the "Connections" editor.



4. In the "Communication drivers" column, select the "SIMATIC S7 200" driver.
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".


Project1 ▸ HMI_2 [KTP1000 Basic PN] ▸ Connections
Connections to S7 PLCs in Devices & Networks
Connections

Name	Communication driver	Station	Partner	Node	Online	C
Verbindung_2	SIMATIC S7 200				☑	
<Add new>						

Parameter
Area pointer
KTP1000 Basic PN
Station

Interface: PROFINET (X1)

HMI device

Address: 192 . 168 . 0 . 2

Access point: S7ONLINE

PLC

Address: 192 . 168 . 0 . 1

Expansion slot: 0

Rack: 0

Cyclic operation: ☑

See the chapter "Parameters for the connection (Page 2534)" for additional details.

10.8.7.3 Parameters for the connection

Cyclic operation

Handling the "Cyclic operation" selection

If "Cyclic operation" is enabled, the HMI device sends a message frame to the CPU at the beginning of communication indicating that certain tags are required on a recurring basis.

The CPU then always transmits the data at the same cyclic interval. This saves the HMI device from having to output new requests for the data.

If cyclic operation is disabled, the HMI device sends a request whenever information is required.

Additional properties:

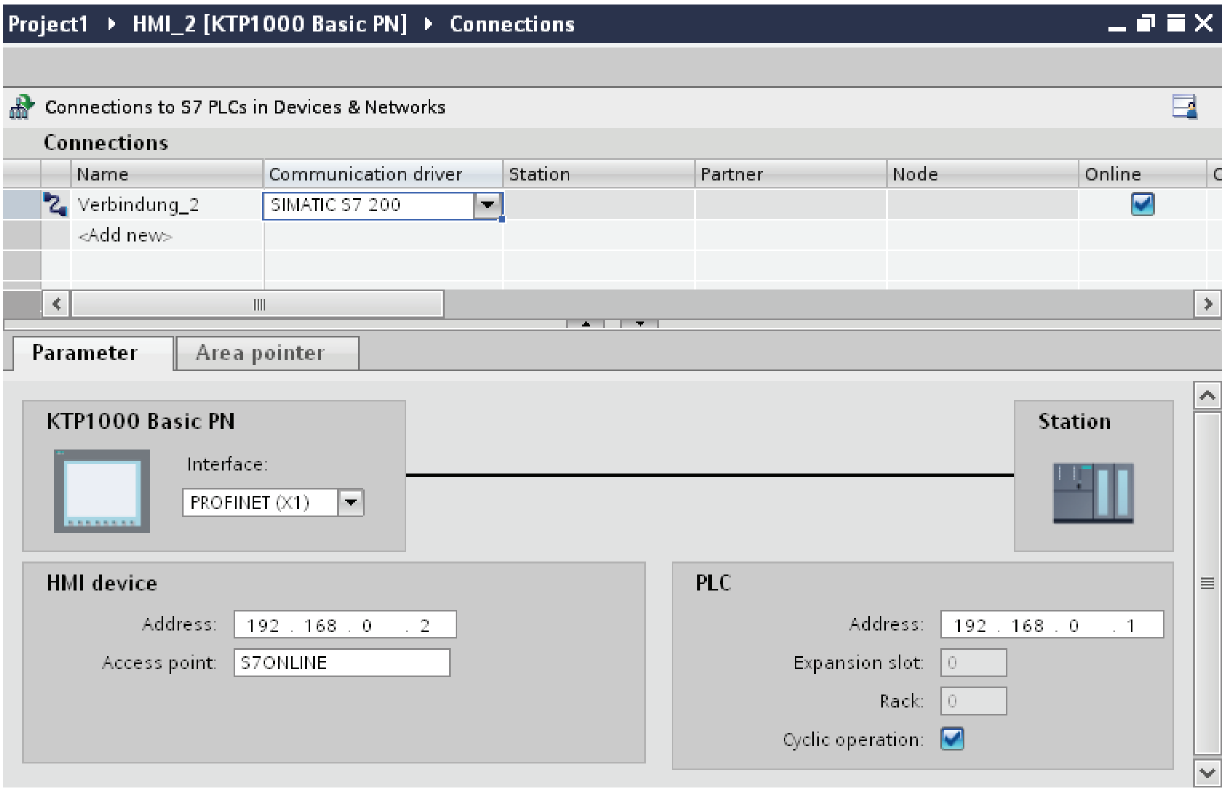
- Cyclic operation reduces data transmission load at the HMI device. The PLC resources are used to relieve load on the HMI device.
- The PLC only supports a certain number of cyclic services. The HMI device handles the operation if the PLC cannot provide any further resources for cyclic services.
- The HMI device generates the cycle if the PLC does not support the cyclic mode.
- Screen tags are not integrated in cyclic operation.
- Cyclic mode is only set up at the restart of Runtime.
- The HMI device transfers several jobs to the PLC if cyclic mode is enabled, depending on the PLC.
- The HMI device only transfers one job to the PLC if cyclic mode is disabled.

Parameters for the connection (SIMATIC S7 200)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device" and "PLC" areas are available for assigning parameters according to the interface used.



Project1 ▸ HMI_2 [KTP1000 Basic PN] ▸ Connections

Connections to S7 PLCs in Devices & Networks

Connections

Name	Communication driver	Station	Partner	Node	Online
Verbindung_2	SIMATIC S7 200				☑
<Add new>					

KTP1000 Basic PN

Interface: PROFINET (X1)

Station

HMI device

Address: 192 . 168 . 0 . 2

Access point: S7ONLINE

PLC

Address: 192 . 168 . 0 . 1

Expansion slot: 0

Rack: 0

Cyclic operation: ☑

Ethernet parameters

Parameters for the HMI device

You set the parameters for the HMI device in the network under "HMI device".. The changes are not transferred automatically to the HMI device. You must change the settings in the Control Panel of the HMI device.

- "Interface"

  If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC.

  Note

  The IP address in the Control Panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

  The IP address already set up in the Control Panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

  The IP address is transferred to the HMI device during project transfer.

  To set up the IP address of the HMI device:

  – Click the HMI device.

  – Open the "Device configuration" editor.

  – Click the Ethernet interface.

  – Assign the IP address in the inspector window under:

    "General > PROFINET interface > Ethernet addresses"

- "Address"

  You assign the IP address of the HMI device in the "Address" area.

  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.

- "Access point"

  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"

  Under "Address", set the IP address of the S7 module to which the HMI device is connected.
- "Expansion slot"

  Defines the number of the expansion slot of the CPU to be addressed.
- "Rack"

  Defines the rack number of the CPU to be addressed.
- "Cyclic mode"

  Note

  The setting "Cyclic operation" cannot be configured for the SIMATIC S7 1200 PLC.

  When cyclic operation is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This increases system performance.

  Disable cyclic mode if you are operating several HMI devices in parallel.

PROFIBUS parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"

  Specifies the physical connection used.
- "Interface"

  For "Interface", you select the HMI device interface via which the HMI device is connected to the PROFIBUS network.
- "Baud rate"

  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"

  You set the PROFIBUS address of the HMI device under "Address". The PROFIBUS address must be unique in the PROFIBUS network.

- "Sole master on bus"

  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master).

  In S7 200, you must set an HMI device as the master.

- "Access point"

  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the network

Under "Network", you set the parameters for the PROFIBUS network to which the HMI device is linked.

- "Profile"

  For "Profile", you select the network profile that is used in the network. In "Profile", set "DP", "Universal", or "Standard". The setting must be identical throughout the network.

- "Highest address"

  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual PROFIBUS address. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud in the OP 73 or the OP 77A, the highest station address must be less than or equal 63.

- "Number of masters"

  For "Number of masters", set the number of masters in the PROFIBUS network. This information is necessary to correctly calculate the bus parameters.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"

  For "Address", set the PROFIBUS address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.

- "Cyclic operation"

Note

The setting "Cyclic operation" cannot be configured for the SIMATIC S7 1200 PLC.

When cyclic operation is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This increases system performance. Disable cyclic mode if you are operating several HMI devices in parallel. This setting is not required for SIMATIC S7-200 PLCs.

MPI parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"

  Specifies the physical connection used.

- "Interface"

  For "Interface", you select the HMI device interface via which the HMI device is connected to the MPI network.

- "Baud rate"

  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"

  For "Address", you set the MPI address of the HMI device. The MPI address must be unique throughout the MPI network.

- "Sole master on bus"

  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master). If you have connected only slaves to the HMI device, you must therefore disable the "Sole master on bus" safety feature.

  In S7 200, you must set an HMI device as the master.

Parameters for the network

Under "Network", you set the parameters for the MPI network to which the HMI device is linked.

- "Profile"

  For "Profile", you select the network profile that is used in the network. In "Profile", set "MPI". The setting must be identical throughout the network.

- "Highest address"

  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual MPI address. The setting must be identical throughout the network.

- "Number of masters"

  This setting is not required for MPI.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"

  For "Address", set the MPI address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.

- "Cyclic mode"

  When cyclic mode is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This improves system performance. Disable cyclic mode if you are operating several HMI devices in parallel. This setting is not required for SIMATIC S7-200.

PPI parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"

  Specifies the physical connection used.

- "Interface"

  For "Interface", you select the HMI device interface via which the HMI device is connected to the PP network.

- "Baud rate"

  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

Note

If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"

  For "Address", you set the PPI address of the HMI device. The PPI address must be unique throughout the PPI network.

- "Access point"

  For "Access point", you set the access point via which the communication partner is reached.

- "Sole master on bus"

  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master). If you have connected only slaves to the HMI device, you must therefore disable the "Sole master on bus" safety feature.

  In S7 200, you must set an HMI device as the master.

Parameters for the network

Under "Network", you set the parameters for the network to which the HMI device is linked.

- "Profile"

  For "Profile", you select the network profile that is used in the network. In "Profile", set "PPI". The setting must be identical throughout the network.

- "Highest address"

  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual MPI address. The setting must be identical throughout the network.

- "Number of masters"

  Set the number of the master on the network to "1".

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"

  For "Address", set the PPI address of the S7 module to which the HMI device is connected.

- "Cyclic operation"

  This parameter is not required for communication via PPI.

10.8.7.4 Data exchange

Data exchange using area pointers

General information on area pointers

Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

Configuration of area pointers

Before you use an area pointer, you enable it under "Connections > Area pointer". You then assign the area pointer parameters.

You can find more detailed information on configuring area pointers in:

Configuring area pointers (Page 2410)

"Screen number" area pointer

Function

The HMI device saves information about the screen called on the HMI device to the "Screen number" area pointer.

This allows the transfer of the current screen contents from the HMI device to the PLC. The PLC can trigger specific reactions such as the call of a different screen.

Application

Configure and enable the area pointer in "Communication > Connections" before you put it into use. You can create only **one** instance of the "Screen number" area pointer and only on **one** PLC.

The screen number is always transferred to the PLC when a new screen is activated or when the focus within a screen changes from one screen object to another.

Structure

The area pointer is a data area in the memory of the PLC with a fixed length of 5 words.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1st word	Current screen type															
2nd word	Current screen number															
3rd word	Reserved															
4th word	Current field number															
5th word	Reserved															

- Current screen type

  "1" for root screen or
  "4" for permanent window
- Current screen number

  1 to 32767
- Current field number

  1 to 32767

"Date/time" area pointer

Function

This area pointer is used to transfer the date and time from the HMI device to the PLC.

The PLC writes control job "41" to the job mailbox.

When it evaluates the control job, the HMI device writes its current date and the time in the data area configured in the "Date/time" area pointer. All definitions are coded in BCD format.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

The date/time data area has the following structure:

Data word	Most significant byte								Least significant byte								
	7							0	7							0	
n+0	Reserved								Hour (0 to 23)								Time
n+1	Minute (0 to 59)								Second (0 to 59)								
n+2	Reserved								Reserved								
n+3	Reserved								Weekday (1 to 7, 1=Sunday)								Date
n+4	Day (1 to 31)								Month (1 to 12)								
n+5	Year (80 to 99/0 to 29)								Reserved								

Note

Note that when you enter the year, values 80 to 99 result in years 1980 through 1999 and the values 0 to 29result in the years 2000 through 2029.

"Date/time PLC" area pointer

Function

This area pointer is used to transfer the date and time from the PLC to the HMI device. Use this area pointer if the PLC is the time master.

The PLC loads the data area of the area pointer. All definitions are coded in BCD format.

The HMI device reads the data cyclically within the configured acquisition cycle and synchronizes itself.

Note

Set an acquisition cycle of sufficient length for the date/time area pointer in order to avoid any negative impact on HMI device performance.
Recommended: Acquisition cycle of 1 minute, if the process allows this.

The date/time data area has the following structure:

DATE_AND_TIME format (in BCD code)

<table>
<tr><th rowspan="2">Data word</th><th colspan="3">Most significant byte</th><th colspan="3">Least significant byte</th></tr>
<tr><th>7</th><th>. . . . . .</th><th>0</th><th>7</th><th>. . . . . .</th><th>0</th></tr>
<tr><td>n+0</td><td colspan="3">Year (80 to 99/0 to 29)</td><td colspan="3">Month (1 to 12)</td></tr>
<tr><td>n+1</td><td colspan="3">Day (1 to 31)</td><td colspan="3">Hour (0 to 23)</td></tr>
<tr><td>n+2</td><td colspan="3">Minute (0 to 59)</td><td colspan="3">Second (0 to 59)</td></tr>
<tr><td>n+3</td><td colspan="3">Reserved</td><td colspan="2">Reserved</td><td>Weekday (1 to 7, 1=Sunday)</td></tr>
<tr><td>n+4 1)</td><td colspan="3">Reserved</td><td colspan="3">Reserved</td></tr>
<tr><td>n+5 1)</td><td colspan="3">Reserved</td><td colspan="3">Reserved</td></tr>
</table>

1) The two data words must exist in the data area to ensure that the data format matches WinCC flexible and to avoid reading false information.

Note

Note that when you enter the year, values 80 to 99 result in years 1980 through 1999 and the values 0 to 29 result in the years 2000 through 2029.

"Coordination" area pointer

Function

The "Coordination" area pointer is used to implement the following functions:

- Detecting the startup of the HMI device in the control program
- Detecting the current operating mode of the HMI device in the control program
- Detecting whether the HMI device is ready to communicate in the control program

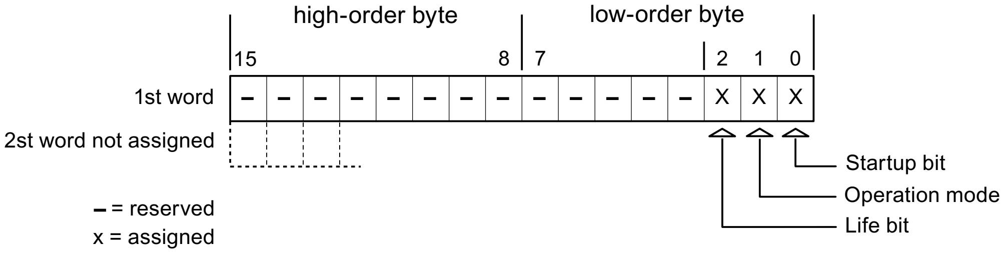
The "Coordination" area pointer has a length of two words.

Use

Note

Each time the area pointer is updated by the HMI device, the entire coordination area is always written.
For this reason, the PLC program must not make any changes in the coordination area.

Assignment of the bits in the "Coordination" area pointer



Startup bit

The startup bit is set briefly to "0" by the HMI device during startup. After startup, the bit is set permanently to "1".

Operating mode

As soon as the HMI device is switched offline by the user, the operating mode bit is set to 1. In normal operation of the HMI device, the state of the operating mode bit is "0". You can find out the current operating mode of the HMI device by querying this bit.

Life bit

The life bit is inverted by the HMI device at intervals of approximately one second. By querying this bit in the PLC program, you can check whether or not the connection to the HMI device still exists.

"Project ID" area pointer

Function

When Runtime starts it can check to see if the HMI device is connected to the correct PLC. This check is important when operating with several HMI devices.

For this, the HMI device compares a value stored on the PLC with the value specified in configuration. This ensures compatibility of the configuration data with the PLC program. If there is no concordance, a system event is given on the HMI device and Runtime is stopped.

Use

To use this area pointer, set up the following during the configuration:

- Define the version of configuration. Possible value between 1 and 255.

  You enter the version in the editor "Runtime settings > General" in the "Identification" area.
- Data address of the value for the version that is stored in the PLC:

  You enter the data address in the editor "Communication > Connections" under "Address".

Connection failure

A connection failure to a device on which the "project ID" area pointer is configured results in all the other connections in the project being switched to "offline".

This behavior has the following requirements:

- You have configured several connections in a project.
- You are using the "project ID" area pointer in at least one connection.

Causes which may set connections "offline":

- The PLC is not available.
- The connection has been switched offline in the engineering system.

"Job mailbox" area pointer

Function

The PLC can use the job mailbox to transfer jobs to the HMI device to trigger corresponding actions on the HMI device. These functions include, for example:

- Display screen
- Set date and time

Data structure

The first word of the job mailbox contains the job number. Depending on the job mailbox, up to three parameters can be transferred.

Word	Most significant byte	Least significant byte
n+0	0	Job number
n+1	Parameter 1	
n+2	Parameter 2	
n+3	Parameter 3	

The HMI device evaluates the job mailbox if the first word of this job is not equal to zero. This means that the parameters must be entered in the job mailbox first, followed by the job number.

When the HMI device accepts the job mailbox, the first word is set to 0 again. The execution of the job mailbox is generally not completed at this point in time.

Job mailboxes

All job mailboxes and their parameters are listed below. The "No." column contains the job number of the job mailbox. Job mailboxes can only be triggered by the PLC when the HMI device is online.

Note

Please note that not all HMI devices support job mailboxes. TP 170A and Micro Panel do not support job mailboxes, for example.

No.	Function	
14	**Setting the time (BCD coded)**	
	Parameter 1	Left byte: - Right byte: hours (0-23)
	Parameter 2	Left byte: minutes (0-59) Right byte: seconds (0-59)
	Parameter 3	-
15	**Set date (BCD code)** [3)]	
	Parameter 1	Left byte: - Right byte: weekday (1-7: Sunday-Saturday)
	Parameter 2	Left byte: day (1-31) Right byte: month (1-12)
	Parameter 3	Left byte: year
23	**User logon**	
	Logs the user on with the name "PLC user" at the HMI device with the group number transferred in parameter 1. The logon is possible only when the transferred group number exists in the project.	
	Parameter 1	Group number 1 to 255
	Parameter 2, 3	-
24	**User logoff**	
	Logs off the current user. (The function corresponds to the "logoff" system function)	
	Parameter 1, 2, 3	-
40	**Transfer date/time to PLC**	
	(in the S7 format DATE_AND_TIME) An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
41	**Transfer date/time to PLC**	
	(In OP/MP format) An interval of at least 5 seconds must be maintained between successive jobs in order to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
46	**Update tags**	
	Causes the HMI device to read the current value of the tags•from the PLC whose update ID matches the value transferred in parameter 1. (Function corresponds to the "UpdateTag" system function.)	
	Parameter 1	1 - 100
49	**Delete alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Warnings" class from the alarm buffer.	
	Parameter 1, 2, 3	-
50	**Delete alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Errors" class from the alarm buffer.	
	Parameter 1, 2, 3	-

No.	Function	
14	Setting the time (BCD coded)	
51	Screen selection [2]	
	Parameter 1	Screen number
	Parameter 2	-
	Parameter 3	Field number
69	Reading data record from PLC [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	0: Do not overwrite existing data record 1: Overwrite existing data record
70	Writing data record from PLC [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	-

1)	Only devices supporting recipes
2)	OP 73, OP 77A and TP 177A HMI devices also execute the "Screen selection" job mailbox if the on-screen keyboard is active.
3)	The weekday is ignored on HMI device KTP 600 BASIC PN.

"Data record" area pointer

"Data record" area pointer

Function

When data records are transferred between the HMI device and PLC, both partners access common communications areas on the PLC.

Data transfer types

There are two ways of transferring data records between the HMI device and PLC:

- Transfer without synchronization
- Transfer with synchronization over the data record

Data records are always transferred directly. That is, the tag values are read from an address or written to an address configured for this tag directly, without redirecting the values by means of interim memory.

Initiating the transfer of data records

There are three ways of triggering the transfer:

- Operator input in the recipe view
- Job mailboxes

  The transfer of data records can also be triggered by the PLC.
- Triggering by configured functions

If the transfer of data records is triggered by a configured function or by a job mailbox, the recipe view on the HMI device remains operable. The data records are transferred in the background.

Simultaneous processing of several transfer requests is, however, not possible. In this case, the HMI device rejects the other transfer requests with a system event.

Sequence of a transfer started by the operator in the recipe view

Reading from the PLC started by the operator in the recipe view

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe number to be read and the status "Transferring" in the data record and sets the data record number to 0.	Abort with system event.
3	The HMI device reads the values from the PLC and displays them in the recipe view. If the recipes have synchronized tags, the values from the PLC are also written to the tags.	
4	The HMI device sets the status "Transfer completed."	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC started by the operator in the recipe view

Step	Action	
	Check: Status word = 0?	
1	Yes	No
	The HMI device enters the recipe and data record number to be written and the status "Transferring" in the data mailbox.	Abort with system event.
2	The HMI device writes the current values to the PLC. If the recipes have synchronized tags, the changed values are synchronized in the recipe view and tags and then written to the PLC.	
3	The HMI device sets the status "Transfer completed."	

Step	Action	
4	If required, the control program can now evaluate the transferred data.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Sequence of the transfer triggered by a job mailbox

The transfer of data records between the HMI device and the PLC can be initiated by the HMI device or by the PLC.

The two job mailboxes No. 69 and No. 70 are available for this type of transfer.

No. 69: Read data record from PLC ("PLC → DAT")

Job mailbox no. 69 transfers data records from the PLC to the HMI device. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	69
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	Do not overwrite existing data record: 0 Overwrite existing data record: 1	

No. 70: Write data record to PLC ("DAT → PLC")

Job mailbox No. 70 transfers data records from the HMI device to the PLC. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	70
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	—	

Sequence when reading from the PLC with job mailbox "PLC → DAT" (no. 69)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device reads the values from the PLC and saves these to the data record defined in the job mailbox.	
4	• If "Overwrite" was selected in the job, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "Do not overwrite" was selected in the job, and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Sequence of writing to the PLC using job mailbox "DAT → PLC" (no. 70)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device fetches the values of the data record specified in the job from the data medium and writes the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Sequence of the transfer when triggered by a configured function

Reading from the PLC using a configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.
3	The HMI device reads the values from the PLC and stores them in the data record specified in the function.	
4	• If "Yes" was selected for the "Overwrite" function, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "No" was selected for the "Overwrite" function and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC by means of configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.
3	The HMI device fetches the values of the data record specified in the function from the data medium and transfers the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Possible causes of error when transferring data records

Possible causes of faults

The section below shows possible causes of errors which lead to a data record transfer being terminated with errors:

- Tag address not set up on the PLC
- Overwriting data records not possible
- Recipe number does not exist
- Data record number does not exist

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Reaction to an aborted transfer due to errors

If the transfer of data records is aborted due to errors, the HMI device reacts as follows:

- Triggering by the operator in the recipe view

  Information in the status bar of the recipe view and output of system events
- Triggered by function

  Output of system events
- Triggering by job mailbox

  No return message on the HMI device

You can nonetheless evaluate the status of the transfer by querying the status word in the data mailbox.

Transfer without synchronization

If you select asynchronous transfer of data records between the HMI device and PLC, there is no coordination over the common data areas. It is therefore unnecessary to set up a data area during configuration.

Asynchronous data record transfer can be a useful alternative, for example, when:

- The system is capable of excluding the risk of uncontrolled overwriting of data by the communication peer.
- The PLC does not require information about the recipe number and data record number.
- The transfer of data records is triggered by the operator of the HMI device.

Reading values

When a read job is triggered, the values are read from the PLC addresses and transferred to the HMI device.

- Triggering by the operator in the recipe view:

  The values are downloaded to the HMI device. You can then process, edit, or save these values, for example.
- Triggering by a function or job mailbox:

  The values are saved immediately to the data volume.

Writing values

When a write job is triggered, the values are written to the PLC addresses.

- Triggering by the operator in the recipe view:

  The current values are written to the PLC.
- Triggering by a function or job mailbox:

  The current values are written to the PLC from the data medium.

Transfer with synchronization

If you select synchronous transfer, both communication partners set status bits in the common data area. You can use this mechanism to prevent uncontrolled overwriting of data in either direction in your control program.

Application

Synchronous data record transfer can be a useful solution, for example, when:

- The PLC is the "active partner" in the transfer of data records.
- The PLC evaluates the information about the recipe number and data record number.
- The transfer of data records is triggered by means of a Job mailbox.

Requirements

In order to synchronize transfer of data records between the HMI device and the PLC, the following requirements must be met during configuration:

- An area pointer has been set up: "Communication > Connections" editor in "Area pointer".
- The PLC with which the HMI device synchronizes transfer of data records is specified in the recipe:

  "Recipes" editor in the Inspector window the "Coordinated transfer of data records" option under "General > Synchronization > Settings".

Structure of the data area

The data area has a fixed length of 5 words. Structure of the data area:

	15		0
1. Word	Current recipe number (1 - 999)		
2. Word	Current data record number (0 - 65535)		
3. Word	Reserved		
4. Word	Status (0, 2, 4, 12)		
5. Word	Reserved		

- Status

  The status word (word 4) can adopt the following values:

Value		Meaning
Decimal	Binary	
0	0000 0000	Transfer permitted, data record free
2	0000 0010	Transferring
4	0000 0100	Transfer completed without error
12	0000 1100	Transfer completed with error

Trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out time-triggered for Basic Panels.

For additional information see:

Configuring trend displays for values from the PLC (Page 2155)

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration.

Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Alarms

Configuring alarms

Configure alarms

Several steps are needed to configure alarms, such as operational messages, error alarms, and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with alarms (Page 2167)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a SIMATIC communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
SIMATIC S7 PLCs	WORD, INT	BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, TIMER

How the bit positions are counted

For connections with a SIMATIC communication driver, the following counting method applies:

How the bit positions are counted	Byte 0								Byte 1							
	Most significant byte								Least significant byte							
In SIMATIC S7 PLCs	7							0	7							0
In WinCC you configure:	15							8	7							0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

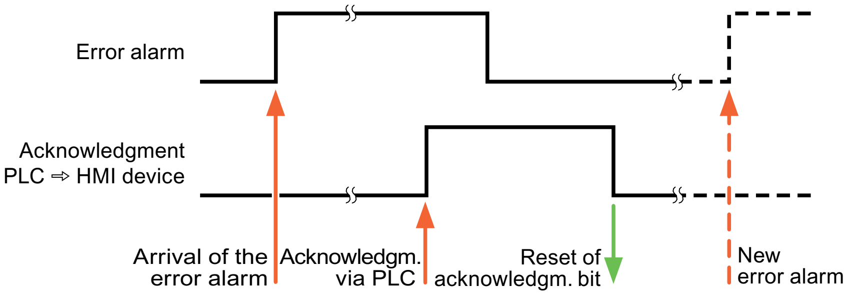
Acknowledgment by the PLC

In "Write acknowledgment tag", you configure the tag or the array tag and the bit number based on which the HMI device can recognize an acknowledgment by the PLC.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In "Read acknowledgment tag", you configure the tag or the array tag and the bit number that is written to the PLC after acknowledgment from the HMI device. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

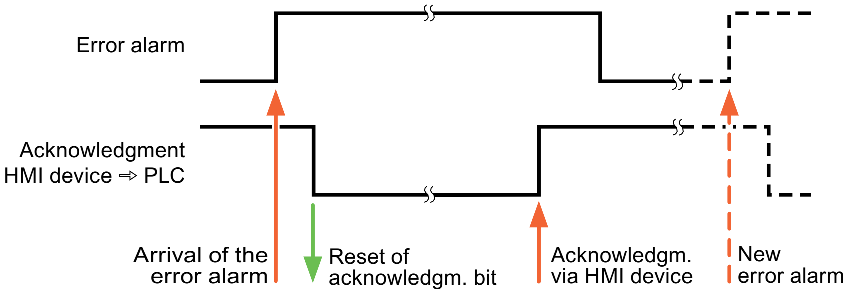
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.


Error alarm
Acknowledgment HMI device ⇨ PLC
Arrival of the error alarm
Reset of acknowledgm. bit
Acknowledgm. via HMI device
New error alarm

10.8.7.5 Performance features of communication

Permitted data types for SIMATIC S7 200

Permitted data types for connections with SIMATIC S7 200

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Length
Bool	1 bit
Byte	1 byte
Char	1 byte
Word	2 bytes
Int	2 bytes
DWord	4 bytes
DInt	4 bytes
Real	4 bytes
StringChar	--
Timer	2 bytes
Array	--

Note

Disconnection with a PPI network

If you are using arrays in the configuration, an array size of approximately 1000 bytes may cause an interruption of the connection.

Use smaller arrays in your configuration.

10.8.8 Communication with other PLCs

10.8.8.1 Communication with other PLCs

Introduction

Communication with other PLCs is communication with PLCs that are not in the SIMATIC family.

These PLCs have proprietary protocols for data exchange. The protocols are configured as communication drivers in WinCC.

Communication drivers

The following communication drivers are supported in WinCC and are already installed:

- Allen-Bradley
  - Allen-Bradley EtherNet/IP
  - Allen-Bradley DF1
- Mitsubishi
  - Mitsubishi MC TCP/IP
  - Mitsubishi FX
- Modicon Modbus
  - Modicon Modbus TCP/IP
  - Modicon Modbus RTU
- Omron
  - Omron Host Link

Communication drivers in WinCC RT Professional

The following communication drivers are supported for RT Professional:

- Allen-Bradley
  - Allen-Bradley EtherNet/IP
- Mitsubishi
  - Mitsubishi MC TCP/IP
- Modicon Modbus
  - Modicon Modbus TCP

Connections between HMI devices and other PLCs

You configure the connections between HMI devices and other PLCs in the "Connections" editor of the HMI device. These connections are non-integrated connections.

10.8.8.2 Distinctive features when configuring

Distinctive features for data exchange

Distinctive features apply when configuring connections to other PLCs, compared to configuring integrated connections.

Note the following distinctive features when configuring:

- Addressing of tags
- Permitted data types
- Distinctive features when configuring area pointers
- Distinctive features when configuring alarms
- Distinctive features when configuring trends

For more detailed information on distinctive features when configuring, refer to Section "Data exchange" of the respective communication driver.

10.8.8.3 Communication drivers

Allen-Bradley

Allen-Bradley communication drivers

Introduction

This section describes the communication between an HMI device and PLCs that use Allen-Bradley communication drivers.

The following communication drivers are supported:

- Allen-Bradley EtherNet/IP
- Allen-Bradley DF1

Data exchange

Data is exchanged by means of tags or area pointers.

- Tags

  The PLC and the HMI device use process values for data exchange. You create tags in the configuration that point to addresses in the PLC. The HMI device reads the value from the defined address, and then displays it. The operator may also enter values on the HMI device, which are then written to the address in the PLC.

- Area pointers

  Area pointers are used to exchange specific data and are only set up when these data are used.

Allen-Bradley EtherNet/IP

Configuring a connection via Allen-Bradley EtherNet/IP

Introduction

You configure a connection to a PLC with an Allen-Bradley EtherNet/IP communication driver in the "Connections" editor of the HMI device.

The Ethernet interfaces are named differently depending on the HMI device.

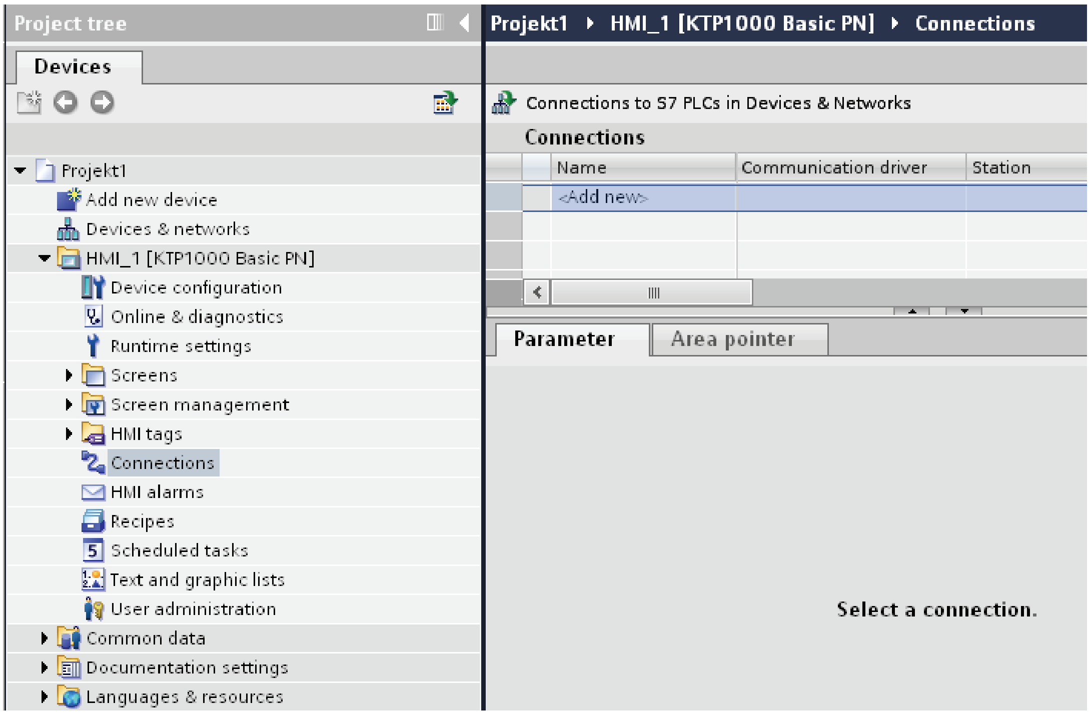
Example: PROFINET interface corresponds to the Ethernet interface

Requirements

- A project is open.
- An HMI device has been created.

Procedure

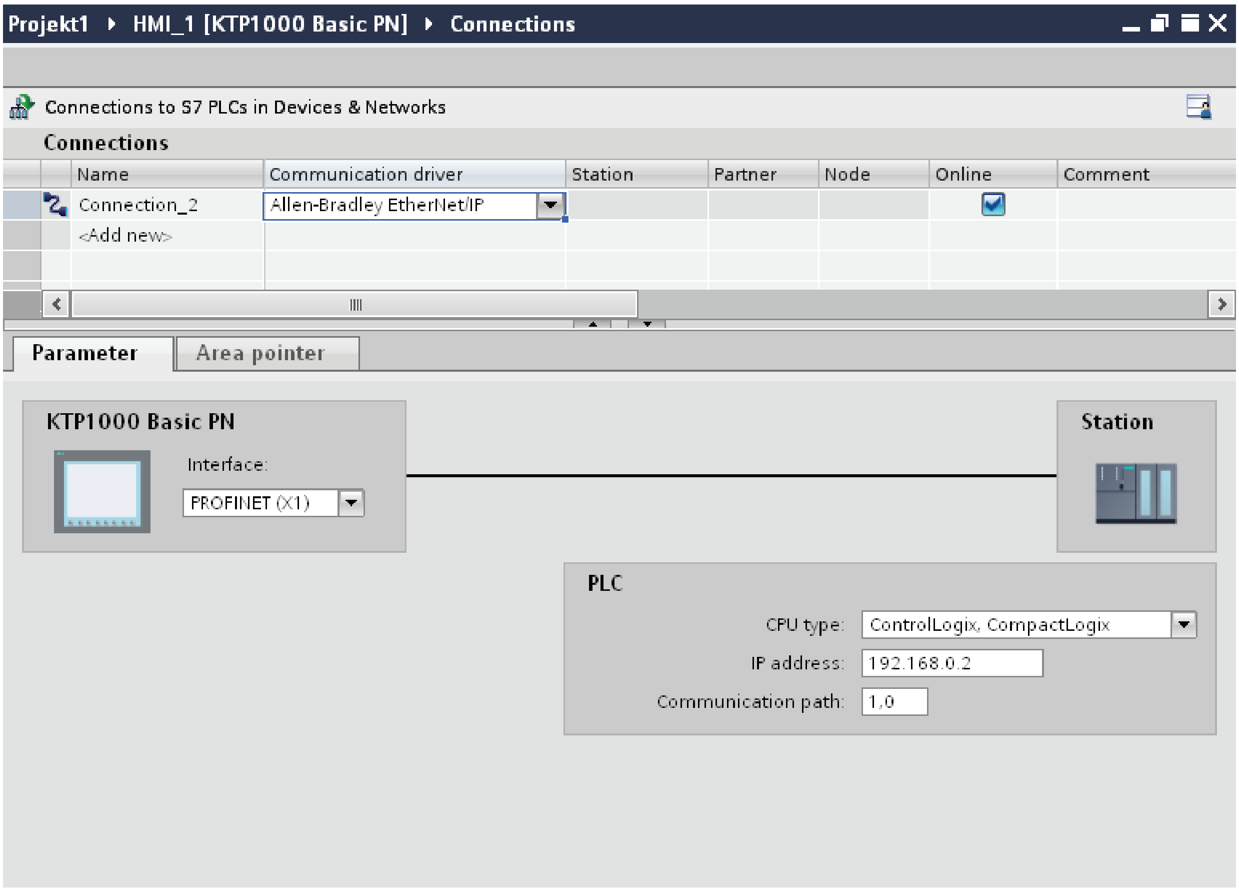
1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.
3. Double-click "<Add>" in the "Connections" editor.



Name	Communication driver	Station
<Add new>		

Select a connection.

4. In the "Communication drivers" column, select the "Allen-Bradley EtherNet/IP" driver.



Projekt1 ▸ HMI_1 [KTP1000 Basic PN] ▸ Connections
Connections to S7 PLCs in Devices & Networks
Parameter
Area pointer
Station

Connections

Name	Communication driver	Station	Partner	Node	Online	Comment
Connection_2	Allen-Bradley EtherNet/IP				☑	
<Add new>						

KTP1000 Basic PN

Interface: PROFINET (X1)

PLC

CPU type: ControlLogix, CompactLogix

IP address: 192.168.0.2

Communication path: 1,0

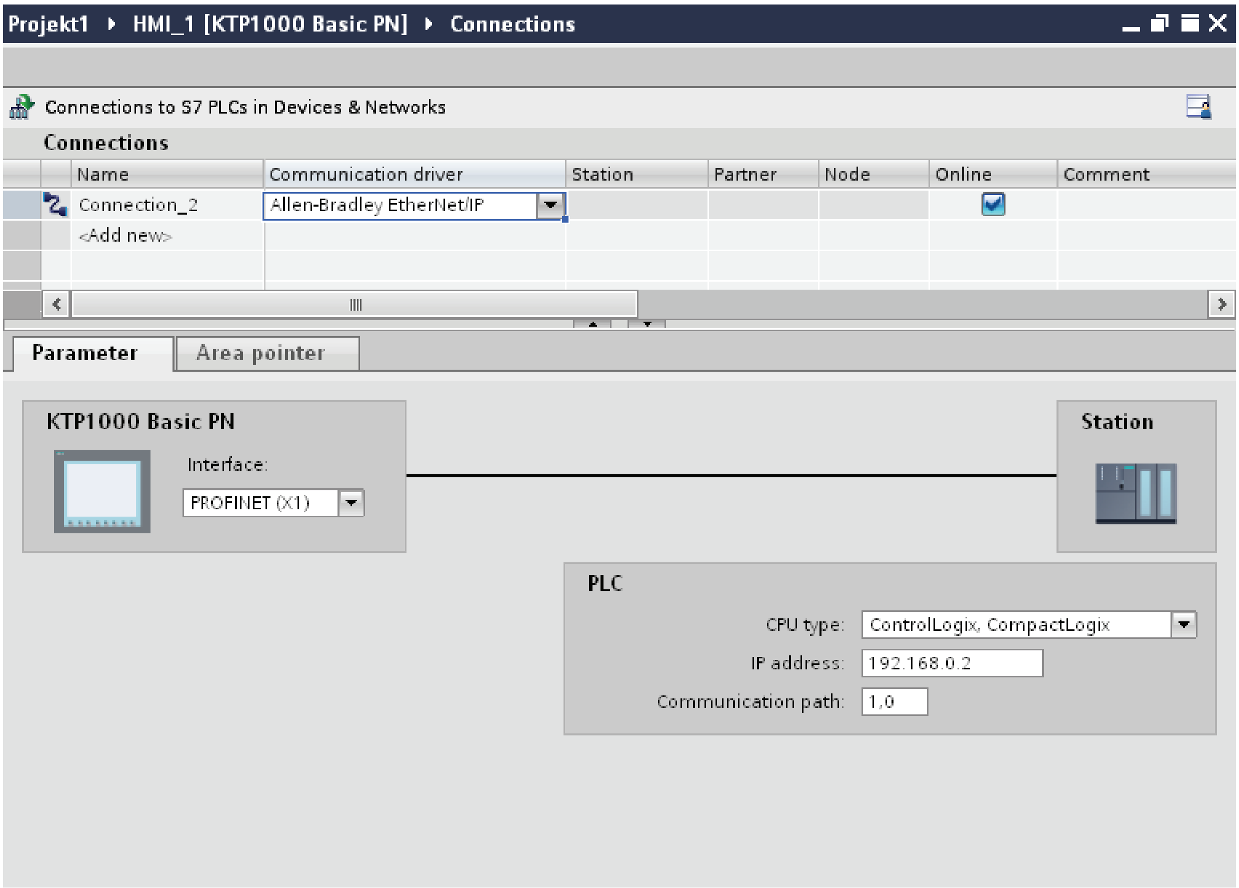
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Allen-Bradley EtherNet/IP)

Parameters to be set

To assign the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "PLC" area is available for assigning parameters according to the interface used.



Projekt1 ▸ HMI_1 [KTP1000 Basic PN] ▸ Connections

Connections to S7 PLCs in Devices & Networks

Connections

Name	Communication driver	Station	Partner	Node	Online	Comment
Connection_2	Allen-Bradley EtherNet/IP				☑	
<Add new>						

Interface: PROFINET (X1)

CPU type: ControlLogix, CompactLogix

IP address: 192.168.0.2

Communication path: 1,0

Parameters for the HMI device

You can select only one interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC. The IP address is transferred to the HMI device upon subsequent loading.

Note

The IP address in the control panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

The IP address already set up in the control panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

To set up the IP address of the HMI device:

1. Click on the HMI device.
2. Open the "Device configuration" editor.
3. Click the Ethernet interface.
4. Assign the IP address in the inspector window under:

   "General > PROFINET interface > Ethernet addresses"

Parameters for the PLC

- CPU type

  For "CPU type", set the CPU type of the PLC used.
- IP address

  Set the IP address or host name of the Ethernet/IP module of the PLC. Only the IP address can be used on a Basic Panel.
- Communication path

  Set the CIP path from the Ethernet module to the PLC. This establishes a logical connection between the Ethernet module and PLC, even if both devices are located in different CIP networks.

  For additional information see: Examples: Communication path

Connecting HMI device to PLC

Connections via Allen-Bradley EtherNet/IP

Connection

The HMI device can be connected to the Allen-Bradley PLC using the following components:

- Existing Ethernet network that also contains the PLCs
- Cross-over Ethernet cable connected directly to the Ethernet interface of the CPU or the communication module

The connection of the HMI device to an Allen-Bradley PLC is limited primarily to the physical connection of the HMI device. Special blocks for the connection are not required in the PLC.

Communication types

Approved communication types with Allen-Bradley EtherNet/IP

The following communication types are system-tested and approved:

- Point-to-point connection to the approved PLCs
- Multipoint connection from a HMI device (Allen-Bradley Ethernet/IP-Client) with up to 4 PLCs with the respectively approved PLCs. CPU types can be mixed.

Connection

Connection with the following PLCs is approved with Allen-Bradley EtherNet/IP:

- CPU type: "ControlLogix, Compact Logix"
  - ControlLogix

    556x(1756-L6x) with Ethernet module 1756-ENBT
  - Guard Logix-System ControlLogix

    556xS(1756-L6xS) with Ethernet module 1756-ENBT
  - CompactLogix
  - 533xE(1769-L3xE) with Ethernet interface onboard
  - 532xE(1769-L2xE) with Ethernet interface onboard
  - 534x (1768-L4x) with Ethernet module 1768-ENBT
- CPU type: "SLC, MicroLogix"
  - MicroLogix 1100 (with Ethernet interface onboard)
  - MicroLogix 1400 (with Ethernet interface onboard)
  - SLC 5/05 (with Ethernet interface onboard)

Performance features of communication

Permitted data types for Allen-Bradley EtherNet/IP

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

CPU type: ControlLogix, CompactLogix

Data type	Length
Bool	1 bit
DInt	4 bytes
Int	2 bytes
Real	4 bytes
SInt	1 byte
String	1 to 80 characters
UDInt	4 bytes
UInt	2 bytes
USInt	1 byte

Permitted data types arrays

Address	Permitted data types
Array	SInt, USInt, Int, UInt, DInt, UDInt, Real
Individual bits from the basic data types of the PLC SInt, USInt, Int, UInt, DInt, UDInt	Bool*

* Any changed value of certain defined bits is written back to the PLC. There is no check to determine whether any other bits have changed. The PLC (or other PLCs) may only read access the value.

CPU type: SLC, MicroLogix

Data type	Operand type	Length
ASCII	A	0 to 80 characters
Bool	N, R, C, T, B, S, I, O	1 bit
DInt	N	4 bytes
Int	N, R, C, T, S	2 bytes
Real	N, F	4 bytes
String	ST	1 to 80 characters
UDInt	N	4 bytes
UInt	N, R, C, T, B, I, O	2 bytes

Note

Strings in RSLogix 5000 have a default length of 82 characters. A maximum of 80 characters can be displayed in WinCC. Always use strings which do not exceed the maximum length of 80 characters.

Permitted data types - arrays

Address	Permitted data types
Array	Int, UInt, DInt, UDInt, Real

Distinctive features for connections with Allen-Bradley Ethernet/IP

With the communication driver Allen Bradley Ethernet/IP and the CPU type SLC, MicroLogix, you can only use array tags for discrete alarms and trends.

Note

I/O modules with 8 or 16 ports occupy one data word on the PLC.

I/O modules with 24 or 32 ports occupy two data words.

The HMI device does not output an error message if using non-existent bits.

You should always make sure that I/O modules with 8 or 24 ports only occupy the bits that are actually assigned to a port.

Supported CPU types for Allen-Bradley EtherNet/IP

CPU types

The following CPU types are supported for configuring the Allen-Bradley EtherNet/IP communication driver.

- CompactLogix
  - 1769-L2xE with Ethernet interface onboard
  - 1769-L3xE with Ethernet interface onboard
  - 1768-L4x with Ethernet module 1768-ENBT
- ControlLogix
  - 1756-L6x with Ethernet module 1756-ENBT
- GuardLogix
  - 1756-L61S with Ethernet module 1756-ENBT
  - 1756-L62S with Ethernet module 1756-ENBT
  - 1756-L63S with Ethernet module 1756-ENBT
- MicroLogix
  - MicroLogix 1100 / 1400
- SLC50x
  - SLC5/05

Addressing in the C.Logix CPU type

Addressing

Addressing

A tag is uniquely referenced in WinCC by means of an address in the PLC. The address must correspond with the tag name in the PLC. The tag address is defined by a string with a length of up to 128 characters.

Using characters for addressing

Valid characters for tag addressing:

- Letters (a to z, A to Z)
- Numbers (0 to 9)
- Underscore (_)

The tag address consists of tag name and other character strings used to specify the tag in the PLC.

Tag name properties:

- The tag name may begin but not end with an underscore character.
- Strings with successive underscore and space characters are invalid.
- The address may not exceed a length of 128 characters.

Note

The characters reserved for tag addressing may not be used in program/tag names or at any other address instance.

The reserved characters are listed below:

Reserved character	Function
.	Element delimiter
:	Definition of a program tag
,	Delimiter for addressing multi-dimensional arrays
/	Reserved for bit addressing.
[ ]	Addressing of array elements or arrays

PLC and program tags

The Allen-Bradley EtherNet/IP communication driver supports addressing of PLC tags (global project tags) and/or program tags (global program tags).

A program tag is declared based on the program name in the PLC and actual tag name which are delimited by colon. PLC tags are simply addressed by their name.

NOTICE

Addressing errors

Addressing errors occur when the tag name and data type are inconsistent.

Note that the tag name defined in the address field in WinCC must match the tag name in the PLC. Make sure that the data types of tags in WinCC match the data types in the PLC.

Note

Module-specific tags, e.g. for data on input and output modules, cannot be addressed directly. Instead, use an alias tag in the PLC.

Example: Local:3:O. Data cannot be addressed in WinCC flexible.

If the alias "MyOut" is defined for Local:3:O in the PLC, you can address with WinCC via MyOut.Data.

Addressing syntax

Notation of addresses

The tables below define the notation for the individual addressing options for Allen-Bradley EtherNet/IP.

Table 10- 12 Access to arrays, basic data types and structure elements

Data types	Type	Address
Basic data types	PLC tag	Tag name
	Program tag	Programname:tagname
Arrays	PLC tag	Array tag
	Program tag	Program name: array tag
Bits	PLC tag	Tagname/bitnumber
	Program tag	Programname:tagname/bitnumber
Structure elements	PLC tag	Structure tag. Structure element
	Program tag	Program name: structure tag. structure element

Note

Bit addressing with the data types Bool, Real and String is not permitted and will cause an addressing fault.

Description of the syntax

Syntax description:

```
(Programname:)tagname([x(,y)(,z)]){.tagname([x(,y)(,z)])}(/bitnumber)
```

- The "( )" defines an optional, single instance of an expression.
- The "{ }" defines an optional expression with multiple single instances.

The address string length may not exceed 128 characters.

Addressing types

Arrays

An array is a data structure that includes a number of data of the same type. WinCC only supports one-dimensional arrays.

In the address column of the tag editor, enter the array name possibly by specifying a start element. The length is defined in the Array Elements input box of the tag editor. If array limits in the PLC are exceeded (due to faulty indexing), addressing errors result.

These arrays must be declared in the PLC as controller or program tags.

Two- or three-dimensional arrays in the PLC can only be addressed in WinCC if these can be mapped area-wise onto one-dimensional arrays .

Note

During all read accesses and all write accesses, all array elements of a tag are always read or written, respectively. The contents of an array tag which is interconnected with a PLC are always transferred whenever there is a change. The HMI device and the PLC cannot concurrently write data to the same array tag for this reason. Instead of writing data only to a single element, the program writes the entire array to the PLC.

Array elements

Elements of one-dimensional, two-dimensional and three-dimensional arrays in the PLC are indexed by setting an index and the corresponding notation in the tag editor. Array addressing starts at element "0", with arrays of all basic types being valid for element addressing. Read/write operations are only carried out at the addressed element, and not for the entire array.

Bits and bit tags

Bit access is allowed to all basic data types with the exception of Bool, Real and String. Bit addressing is also allowed at array/structure elements. The Bool data type is set in WinCC when bits and bit tags in the basic data types are addressed.

Single-digit bit numbers are addressed with "/x" or "/0x" (x = bit number). Bit numbers are defined by up to two digits.

Note

With the "Bool" data type in the data types SInt, Int and DInt, after changing the specified bit the complete tag is then written in the PLC again. In the meantime, no check is made as to whether other bits in the tag have since changed. Therefore, the PLC may have only read access to the specified tag.

Structures

User-defined data types are created by means of structures. These structures group tags of different data types. Structures may consist of basic types, arrays and of other structures. In WinCC, only structure elements are addressed and not entire structures.

Structure elements

Structure elements are addressed by means of the name of the structure and of the required structure element. This addressing is separated by point. In addition to basic data types, the structure elements may represent arrays or other structures. Only one-dimensional arrays may be used as a structure element.

Note

The nesting depth of structures is only limited by the maximum length of 128 characters for the address.

Address multiplexing

Address multiplexing

Address multiplexing is possible with the CompactLogix, ControlLogix CPU type.

Address multiplexing requires two tags:

- "Tag_1" of data type "String"; contains a logical address such as "HMI:Robot5.Block5" as value.

  The value may change to a second valid address, for example, "HMI:Robot4.Block3".
- "Tag_2" is a tag in which the "Allen-Bradley EtherNet/IP" communication driver is set up as a connection.

Enter a valid name of an HMI_tag in square brackets as the address.

- e.g.: "[Tag_1]"
- The tag must be of the String data type.
- The square brackets indicate address multiplexing.
- The address is derived from the actual value in "Tag_1".

Note

You can only multiplex entire Allen-Bradley EtherNet/IP addresses. Multiplexing of address elements is not possible. "HMI:Robot[Tag_1].Block5" is an invalid address.

You can optionally click the arrow right icon in the "Address" column. Replace the "Constant" with the "Multiplex" entry by clicking the arrow on the left edge of the next address dialog box. Now the tag selection list only returns tags of data type "String".

You can also configure a function triggered by a "change of value" event for multiplexed tags.

Examples for addressing

Example of a table for addressing

The table below defines the basic variants for addressing PLC tags. Other addressing variants are possible by means of combination.

Type	Type	Address
General	PLC tag	Tag name
	Program tag	Program:tagname
Array	Access to an element of a 2-dimensional array	Arraytag[Dim1,Dim2]
	Element of structure array (1-dimensional)	Arraytag[Dim1].structureelement
	Bit in element basic type array (2-dimensional)	Arraytag[Dim1,Dim2]/Bit
Structure	Array in structure	Structuretag.arraytag
	Bit in the element of an array in the substructure	Structuretag.structure2.arraytag [element]/bit

Note

Program tags are addressed by leading the address with the program name derived from the PLC with colon delimiter.

Example: Programname:arraytag[Dim1,Dim2]

Access to array elements

Type	Address
PLC tag	Arraytag[Dim1]
	Arraytag[Dim1,Dim2]
	Arraytag[Dim1,Dim2,Dim3]
Program tag	Programname:arraytag[Dim1]
	Programname:arraytag[Dim1,Dim2]
	Programname:arraytag[Dim1,Dim2,Dim3]

Examples: Communication path

Example 1:

Connection with a PLC in the same Allen-Bradley rack.

1,0

Number	Meaning
1	Stands for a backplane connection.
0	Stands for a CPU slot number.

Example 2:

Connection with a PLC in remote Allen-Bradley racks. Two Allen-Bradley racks are networked on Ethernet.

1,2,2,190.130.3.101,1,5

Number	Meaning
1	Backplane connection
2	Stands for the CPU slot number of the second Ethernet module.
2	Stands for an Ethernet connection.
190.130.3.101	IP address of a remote AB rack on the network – in particular the third Ethernet module
1	Backplane connection
5	Slot number of the CPU

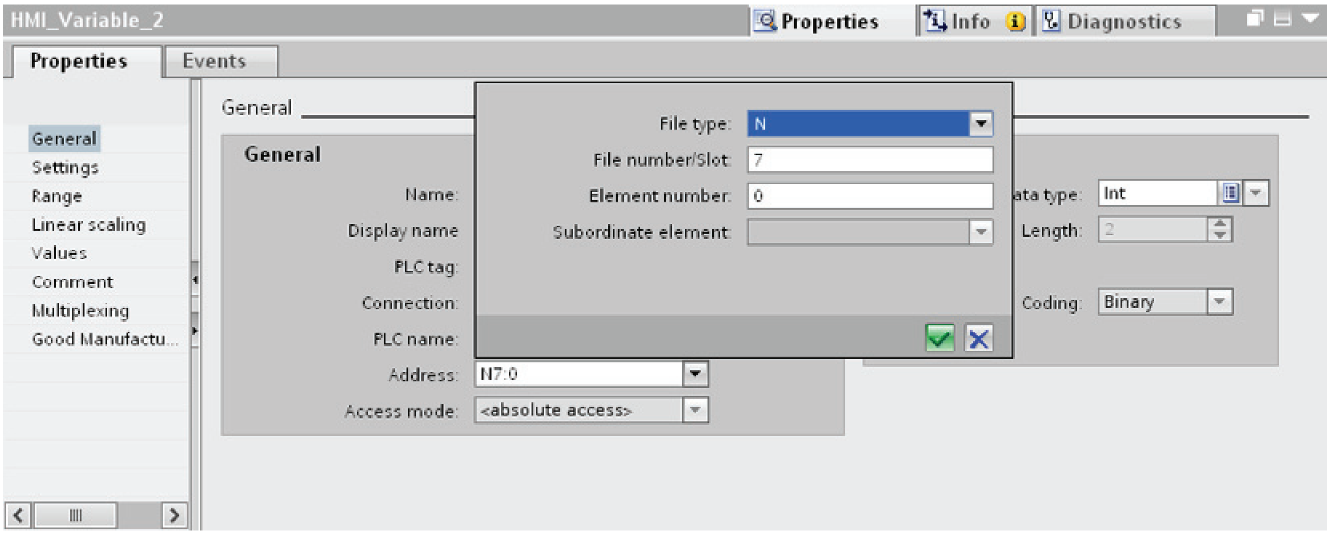
Addressing in the SLC, MicroLogix CPU type

Addressing

Addressing

The addressing in the SLC, MicroLogix CPU type is entered in the following order:

- Operand type
- File number
- Element number
- Child element
- Bit number



File type: N
File number/Slot: 7
Element number: 0
Subordinate element:
Name:
Display name
PLC tag:
Connection:
PLC name:
Address: N7:0
Access mode: <absolute access>
ata type: Int
Length: 2
Coding: Binary

The address then appears in the following format without spaces:

- File type file number : Element number . Child element
- e.g. T8:2.ACC

Operand type

You have the following options under operand type:

- I
- O
- S
- B
- C
- T
- R
- F
- N
- ST
- A

File number

Select the number between two limits under file number:

- Low limit
- High limit

The limit values depend on the selected operand type.

Child element

You can select a child element when you have selected one of the following operand types:

- R
- C
- T

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.

   The project is compiled automatically.

   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Allen-Bradley DF1

Configuring a connection via Allen-Bradley DF1

Introduction

You configure a connection to a PLC with an Allen-Bradley DF1 communication driver in the "Connections" editor of the HMI device.

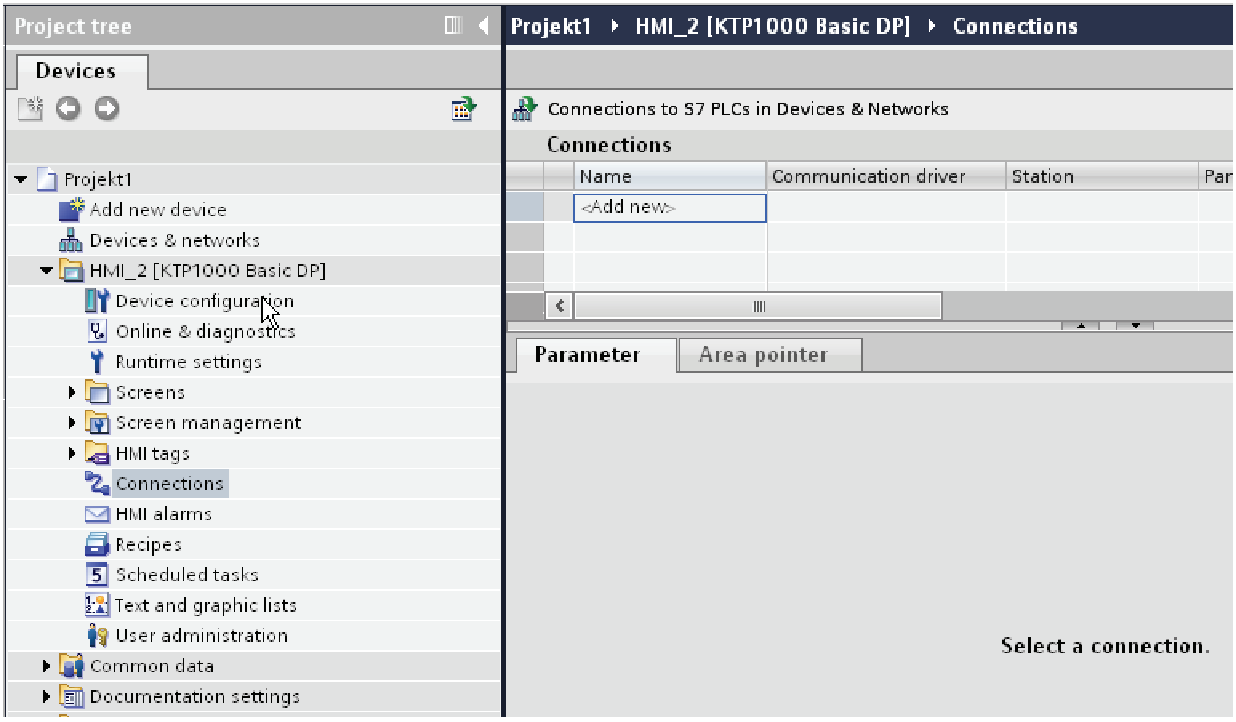
The interfaces are named differently depending on the HMI device.

Requirements

- A project is open.
- An HMI device has been created.

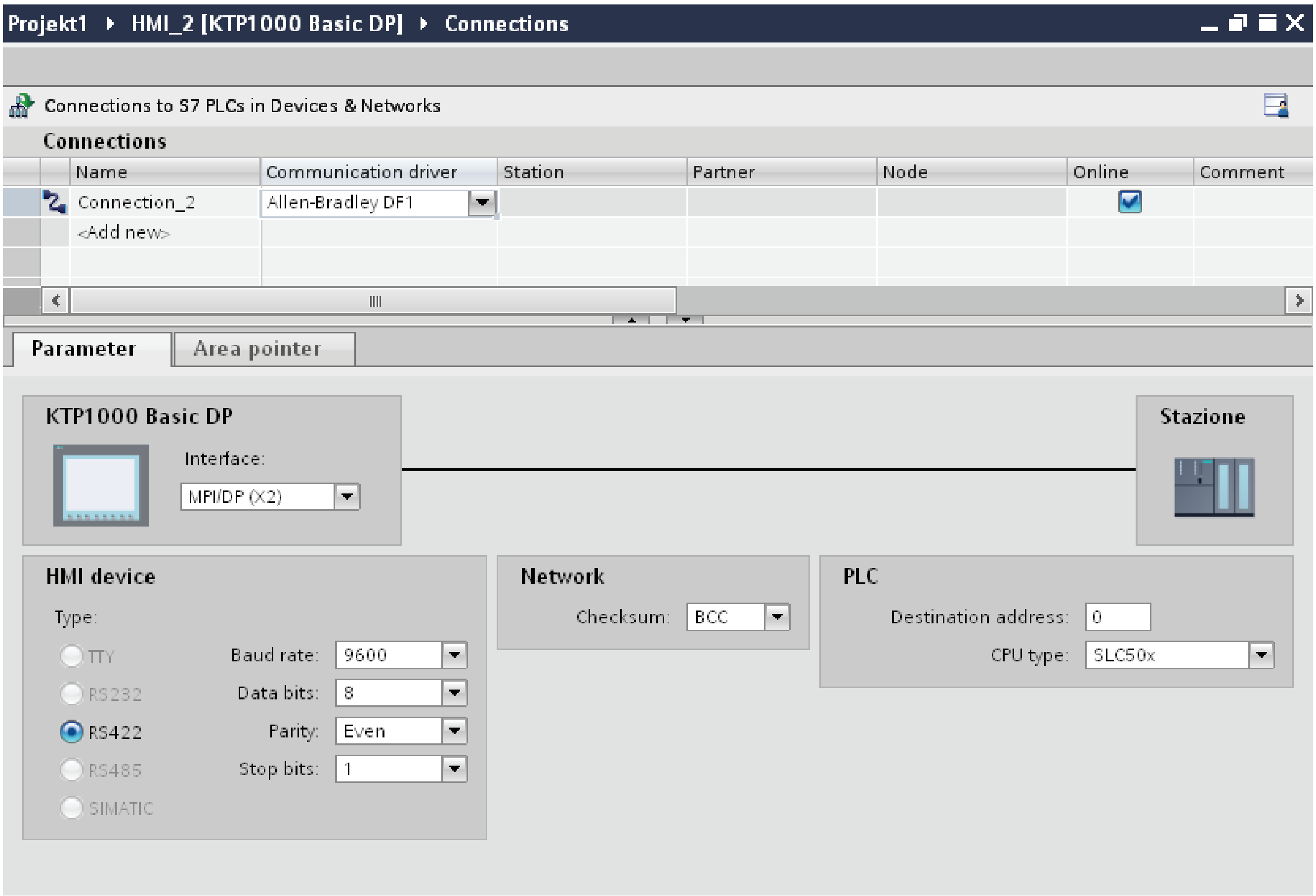
Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.
3. Double-click "<Add>" in the "Connections" editor.



Select a connection.

4. In the "Communication drivers" column, select the "Allen-Bradley DF1" driver.



Projekt1 ▸ HMI_2 [KTP1000 Basic DP] ▸ Connections

Connections to S7 PLCs in Devices & Networks

Connections

	Name	Communication driver	Station	Partner	Node	Online	Comment
	Connection_2	Allen-Bradley DF1				☑	
	<Add new>						

Parameter | Area pointer

KTP1000 Basic DP

Interface: MPI/DP (X2)

Stazione

HMI device

Type:

○ TTY
○ RS232
● RS422
○ RS485
○ SIMATIC

Baud rate: 9600
Data bits: 8
Parity: Even
Stop bits: 1

Network

Checksum: BCC

PLC

Destination address: 0
CPU type: SLC50x

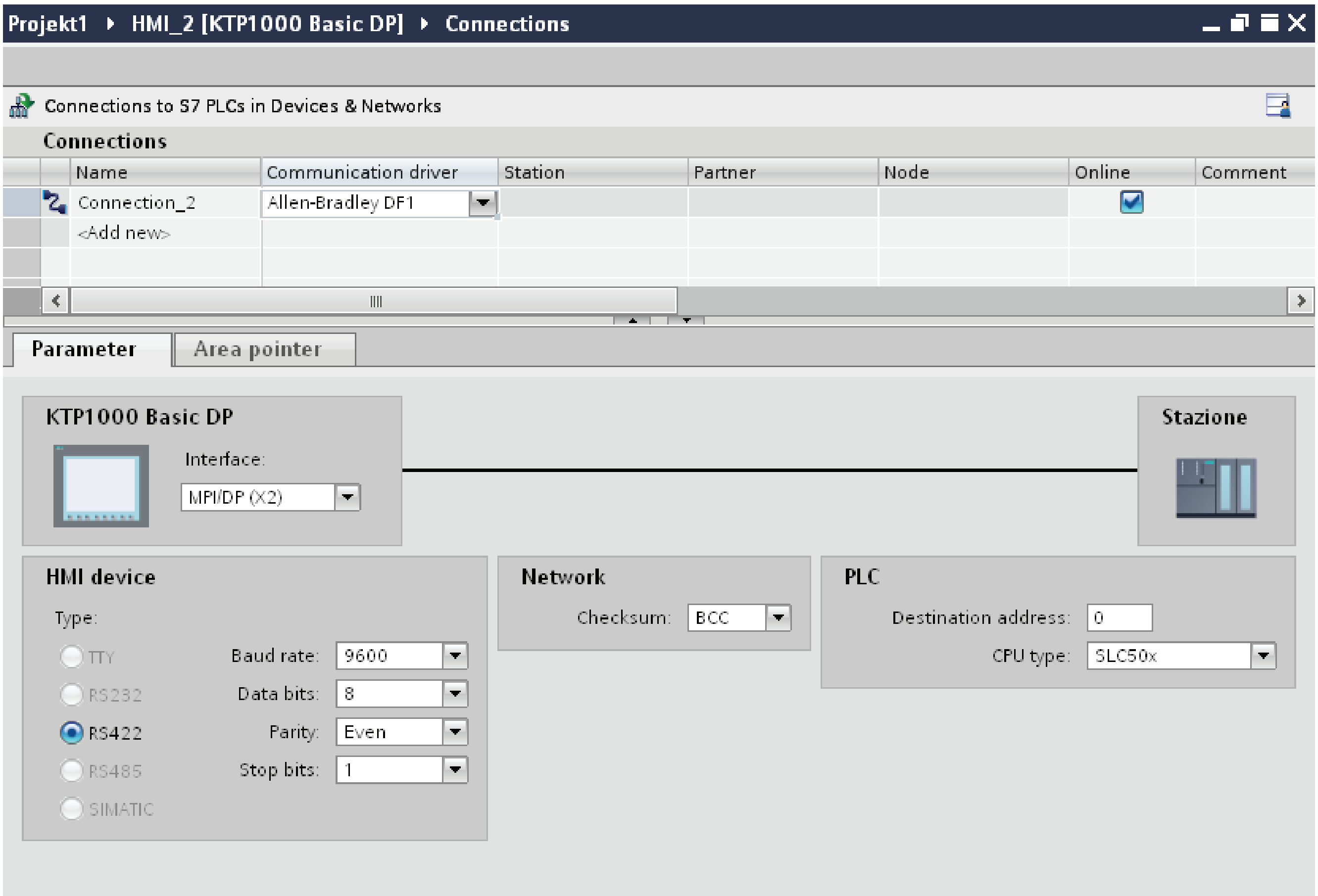
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Allen-Bradley DF1)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device", "Network", and "PLC" areas are available for assigning parameters according to the interface used.



Projekt1 ▸ HMI_2 [KTP1000 Basic DP] ▸ Connections
Connections to S7 PLCs in Devices & Networks
Connections

Name	Communication driver	Station	Partner	Node	Online	Comment
Connection_2	Allen-Bradley DF1				☑	
<Add new>						

Parameter
Area pointer
KTP1000 Basic DP
Stazione

Interface: MPI/DP (X2)

HMI device

Type:

○ TTY
○ RS232
● RS422
○ RS485
○ SIMATIC

Baud rate: 9600
Data bits: 8
Parity: Even
Stop bits: 1

Network

Checksum: BCC

PLC

Destination address: 0
CPU type: SLC50x

Parameters for the HMI device

- Interface

  Under "Interface", you select the interface of the HMI device to which the PLC is connected.

  For additional information, refer to the device manual for the HMI device.

- Type

  Specifies the physical connection used.

  Note

  If you are using the IF1B interface, you must also reconnect the RS-485 receive data and the RTS signal to the rear of the HMI devices via 4 DIP switches.

- Baud rate

  For "Baud rate", select the transmission speed between the HMI device and PLC.

- Data bits

  For "Data bits", you can choose between "7 bits" and "8 bits".

- Parity

  For "Parity", you can choose from "None", "Even", and "Odd".

- Stop bits

  For "Stop bits", you can choose between 1 and 2 bits.

Parameters for the network

- Checksum

  For "Checksum", choose the method for determining the error code: "BCC" or "CRC".

Parameters for the PLC

- Destination address

  For "Destination address", choose the PLC address. If there is a point-to-point DF1 connection, you set the address "0".

- CPU type

  For "CPU type", set the CPU type of the PLC used.

  Note

  Assign the DF1 FULL-DUPLEX driver in the CPU as follows: "NO HANDSHAKING" for "Control Line" and "AUTO-DETECT" for "Embedded Responses".

Connecting HMI device to PLC

Connections via Allen-Bradley DF1

Connection

The connection is established when you have matched the parameters of the PLC and the HMI device. Special blocks for the connection are not required in the PLC.

Note

Rockwell offers a variety of communication adapters for integrating "DF1 devices" for the DH485, DH, and DH+ networks. Of these connections, the direct connection and the connection via KF2 and KF3 module are approved. None of the other connections have been system-tested by SIEMENS AG and are therefore not approved.

Communication partners for Allen-Bradley DF1

Connectable PLCs

The communication drivers listed below support Allen-Bradley PLCs: :

PLC	DF1 (point-to-point) RS-232	DF1 (point-to-point) RS-422	DF1 (multipoint) over KF2 module to DH+ LAN RS-232/RS-422	DF1 (multipoint) over KF3 module to DH485 LAN RS-232
SLC500	–	–	–	X
SLC501	–	–	–	X
SLC502	–	–	–	X
SLC503	X	–	–	X
SLC504	X	–	X	X
SLC505	X	–	–	X
MicroLogix	X	–	–	X
PLC-5 [1]	X	X	X	–

[1] Processors released for PLC-5: PLC-5/11, PLC-5/20, PLC-5/30, PLC-5/40, PLC-5/60, and PLC-5/80.

Communication types

PLCs with Allen-Bradley DF1 communication driver

The communication between the HMI device and the following Allen Bradley PLCs is described in this section:

- SLC500
- SLC501
- SLC502
- SLC503
- SLC504
- SLC505
- PLC5
- MicroLogix

In these PLCs the connection is made by the PLC-internal protocols Allen Bradley DF1, Allen Bradley DH485 and Allen Bradley DH+.

The Allen-Bradley DF1 communication driver is used here, the protocol of which is converted into one of the other two PLC-internal protocols in multipoint communication with the communication modules KF2 (Allen Bradley DH+) and KF3(Allen Bradley DH485).

Enabled types of communication with Allen-Bradley DF1

The following communication types are system-tested and enabled:

- HMI (Allen Bradley DF1)

  Point-to-point connection
- HMI (Allen Bradley DF1)

  Via KF2 module to Allen Bradley DH+ (communication with up to 4 PLCs)
- HMI (Allen Bradley DF1)

  Via KF3 module to Allen Bradley DH485 (communication with up to 4 PLCs)

Connectable PLCs

The Allen Bradley DF1 communication driver is available for the following Allen-Bradley PLCs:

PLC	DF1 (point-to-point) RS 232	DF1 (point-to-point) RS 422	DF1 (multipoint) via KF2 module to DH+ LAN RS 232/RS 422	DF1 (multipoint) via KF3 module to DH485 LAN RS 232 [2]
SLC500	–	–	–	X
SLC501	–	–	–	X
SLC502	–	–	–	X
SLC503	X [2]	–	–	X
SLC504	X [2]	–	X	X
SLC505	X [2]	–	–	X
MicroLogix	X [2]	–	–	X
PLC-5 [1]	X	X	X	–

[1] Only the following processors are approved for PLC-5: PLC-5/11, PLC-5/20, PLC-5/30, PLC-5/40, PLC-5/60 und PLC-5/80.

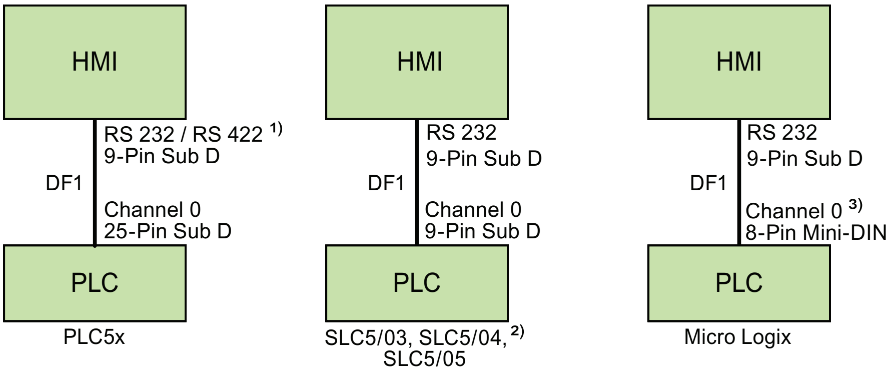
[2] For HMI devices which only have an RS 422/485 interface and the communication partner is an RS 232 interface, the RS 422/232 converter is tested and approved.

Order number: 6AV6 671-8XE00-0AX0

DF1 protocol with multi-point connection

Point-to-point connection with DF1 protocol

Only point-to-point connections can be established with the DF1 protocol.



1) Only RS 232 is possible for Panel PC and PC.

2) A point-to-point connection to the SLC500, SLC501, and SLC502 PLCs via DF1 is not possible.

3) For MicroLogix ML1500 LRP, Channel 1 (9-pin Sub D) is also possible.

Connecting cable

HMI panel interface used	For connection to PLC5x	For connection to SLC5/03, SLC5/04, SLC5/05	For connection to MicroLogix
RS 232 9-pin	Allen-Bradley cable 1784-CP10	Allen-Bradley cable 1747-CP3	Allen-Bradley cable 1761-CBL-PM02
RS 422 9-pin	Connecting cable 9-pin Sub D RS 422	—	—

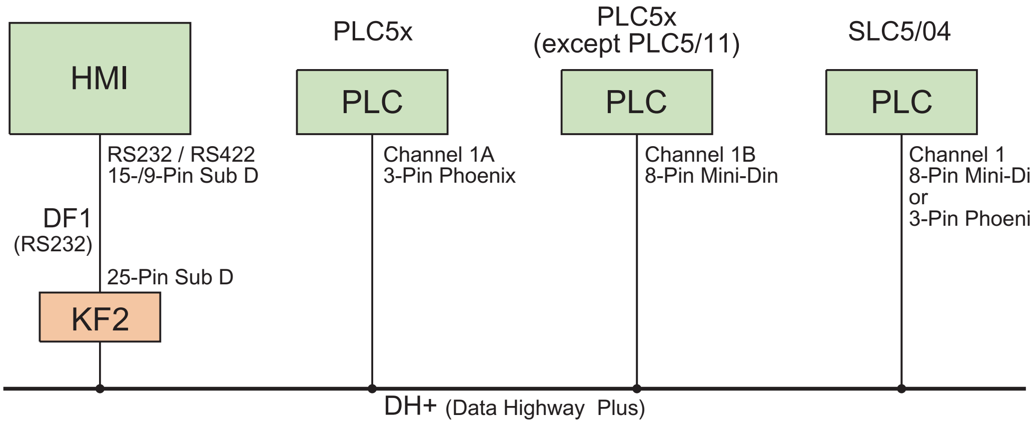
Refer to the relevant device manual to determine which HMI device interface is to be used.

The cable pin assignments can be found in Section "Connecting cables for Allen-Bradley".

DF1 protocol with multi-point connection via KF2 module

DF1 protocol with multi-point connection via KF2 module to DH+ LAN

The use of a KF2 protocol interface module enables a connection to be made to PLCs in the DH+ LAN (Data Highway Plus Local Area Network).



Connecting cable

HMI panel interface used	For connection to KF2 interface module
RS 232 9-pin	Allen-Bradley cable 1784-CP10 and 25-pin socket/socket adapter
RS 422 9-pin	9-pin Sub D RS 422 connecting cable and 25-pin female/female adapter

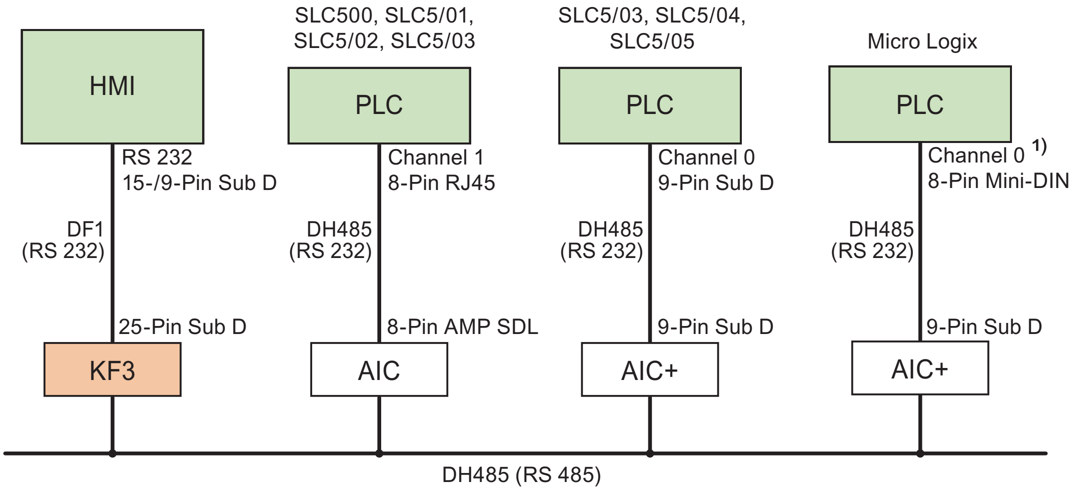
Refer to the Allen-Bradley documentation for the cable connection from the PLCs to the DH+ data bus.

Refer to the relevant device manual to determine which HMI device interface is to be used.

The cable pin assignments can be found in Section "Connecting cables for Allen-Bradley".

DF1 protocol with multi-point connection via KF3 module

DF1 protocol with multi-point connection via KF3 module to DH485 LAN



1) For MicroLogix ML1500 LRP, Channel 1 (9-pin Sub D) is also possible.

Connecting cable

HMI panel interface used	For connection to KF3 interface module
RS 232 9-pin	Allen-Bradley cable 1784-CP10 and 25-pin socket/socket adapter

Refer to the relevant device manual to determine which HMI device interface is to be used.

The cable pin assignments can be found in Section "Connecting cables for Allen-Bradley".

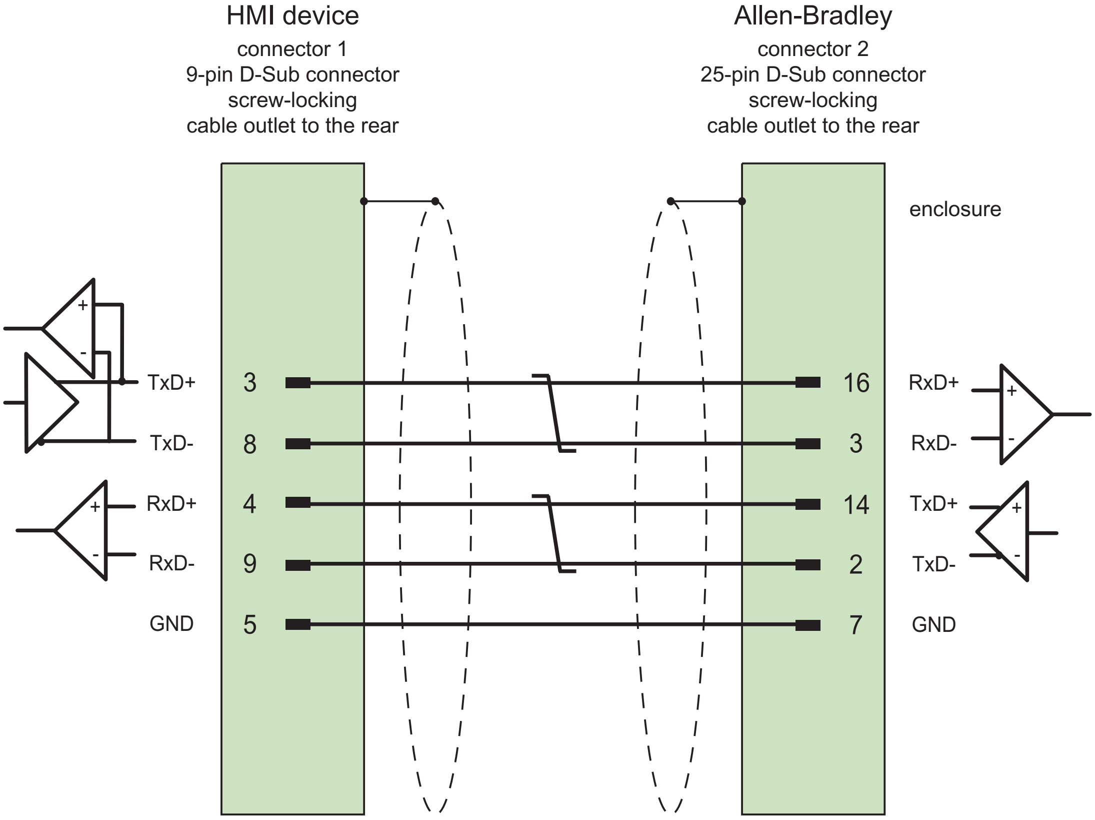
Connecting cables for Allen-Bradley DF1

Connecting cable 9-pin Sub D RS 422 for Allen-Bradley

Connecting cable 9-pin Sub D RS 422

For interconnecting the HMI device (RS 422, 9-pin sub D) - PLC5x, KF2, KF3

You require an additional 25-pin, female / female adapter (gender changer) for interconnections with KF2 and KF3.



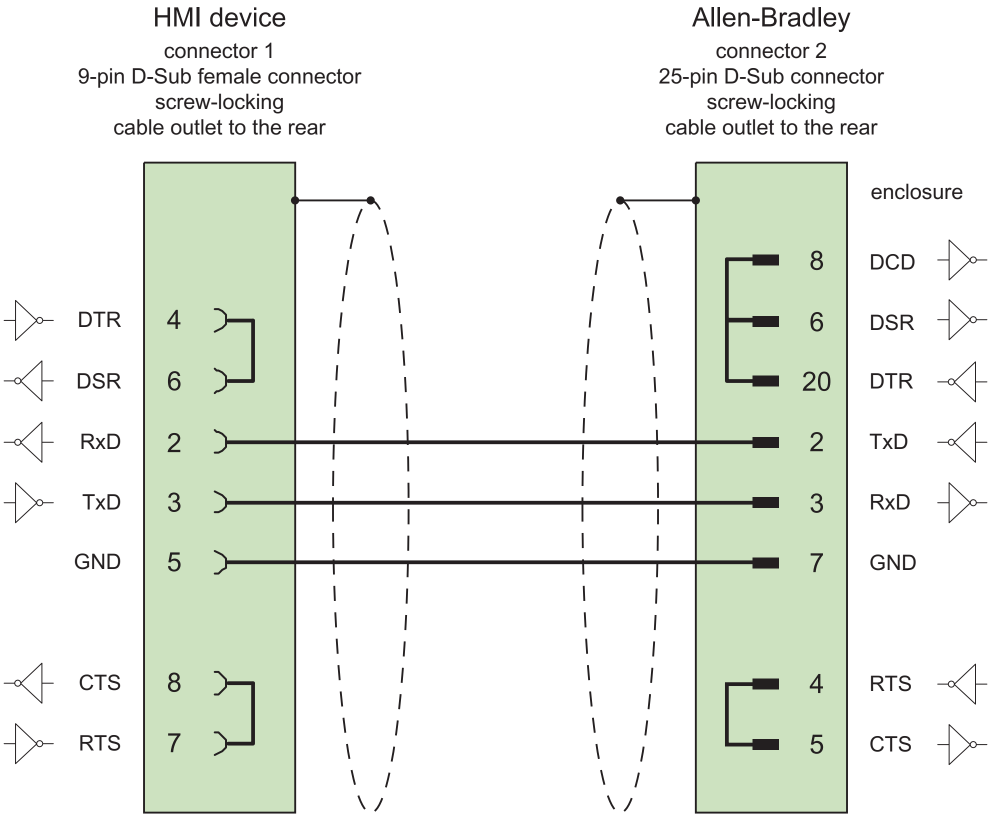
Shield with large-area contact to housing at both ends, interconnected shield contacts
Cable: 3 x 2 x 0.14 mm², shielded,
max. length 60 m

Connecting cable 1784-CP10, RS 232, for Allen-Bradley

Allen-Bradley cable 1784-CP10

For interconnecting the HMI device (RS 232, 9-pin sub D) - PLC5x, KF2, KF3

You require an additional 25-pin, female / female adapter (gender changer) for interconnections with KF2 and KF3.



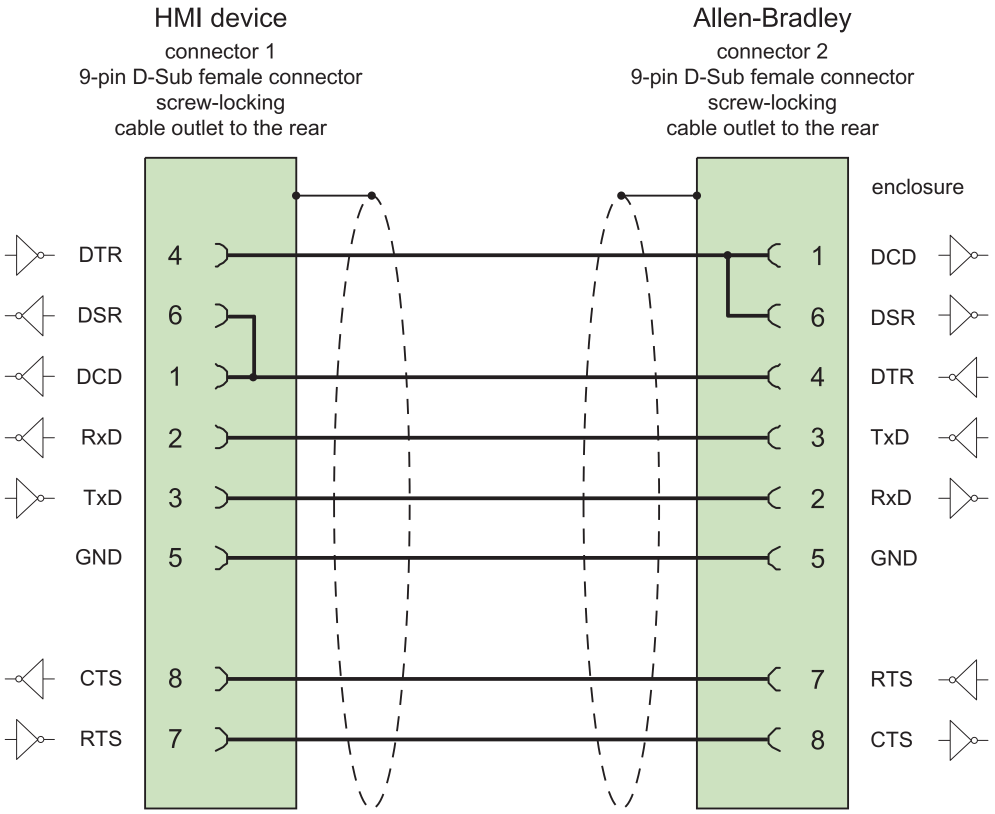
Screen connected with housing over large area on both sides

max. length 15 m

Connecting cable 1747-CP3, RS-232, for Allen-Bradley

Allen-Bradley cable 1747-CP3

For interconnecting the HMI device (RS 232, 9-pin sub D) - SLC503, SLC504, SLC505 (Channel 0), AIC+



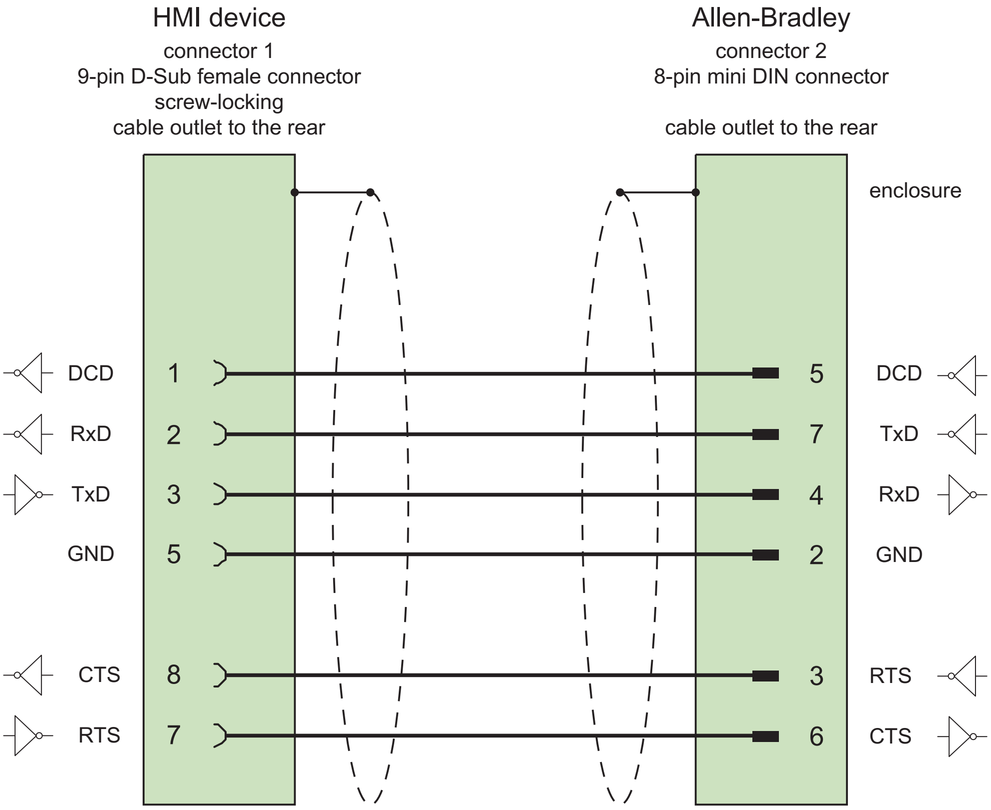
Screen connected with housing over large area on both sides

max. length 3 m

Connecting cable 1761-CBL-PM02, RS-232, for Allen-Bradley

Allen-Bradley cable 1761-CBL-PM02

For interconnecting the HMI device (RS 232, 9-pin sub D) - Micro Logix, AIC+



Screen connected with housing over large area on both sides

max. length 15 m

Performance features of communication

Permitted data types for Allen-Bradley DF1

Permitted data types for Allen-Bradley DF1

The table lists the user data types that can be used when configuring tags and area pointers.

Data type	Operand type	Length
ASCII	A [1]	1 to 80 characters
Bool	N, R, C, T, B, S, I, O	1 bit
Int	N, R, C, T, S	2 bytes
DInt	N, D [2]	4 bytes
UInt	N, R, C, T, B, I, O, D [2]	2 bytes
UDInt	N, D [2]	4 bytes
Real	N, F [1]	4 bytes

[1] Selectable depending on the selected CPU type.

[2] Only for PLC5 CPU type

Abbreviations

In WinCC, formats of the data types are abbreviated as follows:

- UNSIGNED INT = UInt
- UNSIGNED LONG = UDInt
- SIGNED INT = Int
- SIGNED LONG = DInt

Distinctive features for connections with Allen-Bradley DF1

With Allen Bradley DF1, array tags may only be used for discrete alarms and trends.

Note

I/O modules with 8 or 16 ports occupy one data word on the PLC.

I/O modules with 24 or 32 ports occupy two data words.

The HMI device does not output an error message if using non-existent bits.

You should always make sure that I/O modules with 8 or 24 ports only occupy the bits that are actually assigned to a port.

Supported CPU types for Allen-Bradley DF1

CPU types

The following CPU types are supported for configuring the Allen-Bradley DF1 communication driver.

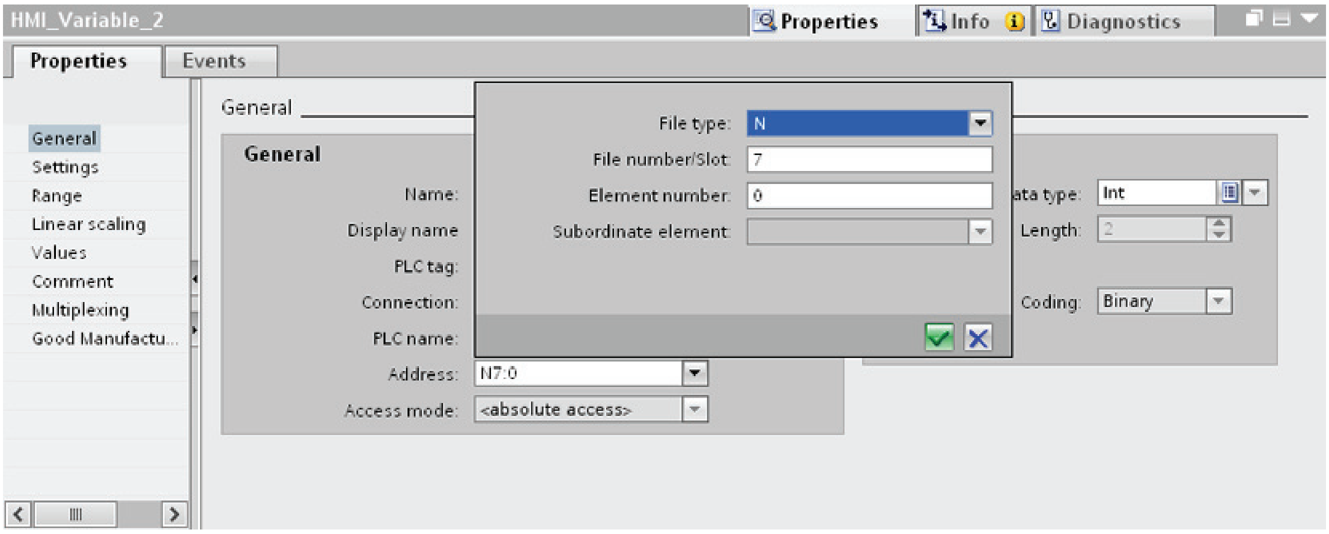
- SLC
  - SLC500
  - SLC501
  - SLC502
  - SLC503
  - SLC504
  - SLC505
- MicroLogix
  - MicroLogix 1x00
  - MicroLogix 1100 / 1400
- PLC 5
  - PLC-5/11
  - PLC-5/20
  - PLC-5/40
  - PLC-5/60
  - PLC-5/80

Addressing

Addressing

The addressing is entered in the following order in the Allen-Bradley DF1 communication driver:

- Operand type
- File number
- Element number
- Child element
- Bit number



Name:
Display name
PLC tag:
Connection:
PLC name:
Address: N7:0
Access mode: <absolute access>

File type: N
File number/Slot: 7
Element number: 0
Subordinate element:

ata type: Int
Length: 2
Coding: Binary

The address then appears in the following format without spaces:

- File type file number : Element number . Child element
- e.g. T8:2.ACC

Operand type

You have the following options under operand type:

- I
- O
- S
- B
- T
- C
- R
- N
- A
- D only for PLC5 CPU type

File number

Select the number between two limits under file number:

- Low limit
- High limit

The limit values depend on the selected file type.

Child element

You can select a child element when you have selected one of the following data types:

- R
- C
- T

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   – Interface
   – Transfer parameters
   – Target storage location
3. Start the project transfer.

   The project is compiled automatically.

   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Data exchange

Area pointers for Allen-Bradley

Area pointers for connections using an Allen-Bradley communication driver

You use an area pointer to access a data area in the PLC.

For more detailed information on area pointers and their configuration, refer to Section "Data exchange using area pointers".

Distinctive features for connections via Allen-Bradley EtherNet/IP

You can configure the following area pointers

Area pointer	Allen-Bradley EtherNet/IP	Allen-Bradley DF1
Screen number	Yes	Yes
Date/time	Yes	Yes
Date/time PLC	Yes	Yes
Coordination	Yes	Yes
Project ID	Yes	Yes
Job mailbox	Yes	Yes
Data record	Yes	Yes

Restrictions Allen-Bradley Ethernet/IP

The following restrictions apply for configuring area pointers.

CPU type	Data types	File types
ControlLogix,CompactLogix	Int, UInt	--
SLC, MicroLogix	Int, UInt	N, B

Restrictions Allen-Bradley DF1

The following restrictions apply for configuring area pointers.

CPU type	Data types	File types
MicroLogix	--	N, O, I, B
SLC50x	--	N, O, I, B
PLC5	--	N, O, I, B

See also

Data exchange using area pointers (Page 2676)

Trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out time-triggered for Basic Panels.

For additional information see:

Configuring trend displays for values from the PLC (Page 2155)

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration.

Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Alarms

Configuring alarms

Configuring alarms for non-integrated connections

Several steps are necessary to configure alarms such as warnings, error messages and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with alarms (Page 2167)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Restrictions

Only tags whose "File type" is "N", "O", "I", "S" and "B" are allowed for use as a "trigger tag" for discrete alarms. These tags are only valid for the data types "Int" and "UInt".

Data types

For connections with an Allen-Bradley communication driver, the following data types are supported:

Communication drivers	PLC	Permitted data types	
		Discrete alarms	Analog alarms
Allen-Bradley DF1	SLC500, SLC501, SLC502, SLC503, SLC504, SLC505, PLC5, MicroLogix	Int, UInt	Int, UInt, Long, ULong, Real
Allen-Bradley EtherNet/IP	ControlLogix, CompactLogix, SLC, Micrologix	Int, UInt	SInt, USInt, Int, UInt, DInt, UDInt, Real

How the bit positions are counted

For connections with an Allen-Bradley communication driver, the following counting method applies:

How the bit positions are counted	Left byte								Right byte							
In Allen-Bradley PLCs	15							8	7							0
In WinCC you configure:	15							8	7							0

See also

Alarm system in WinCC (Page 2158)

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

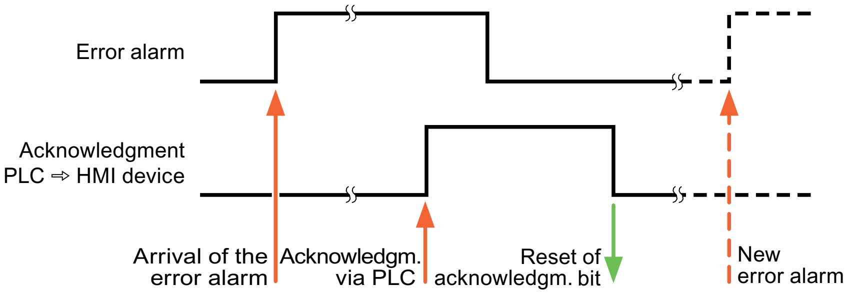
Acknowledgment by the PLC

In "Write acknowledgment tag", you configure the tag or the array tag and the bit number based on which the HMI device can recognize an acknowledgment by the PLC.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In "Read acknowledgment tag", you configure the tag or the array tag and the bit number that is written to the PLC after acknowledgment from the HMI device. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

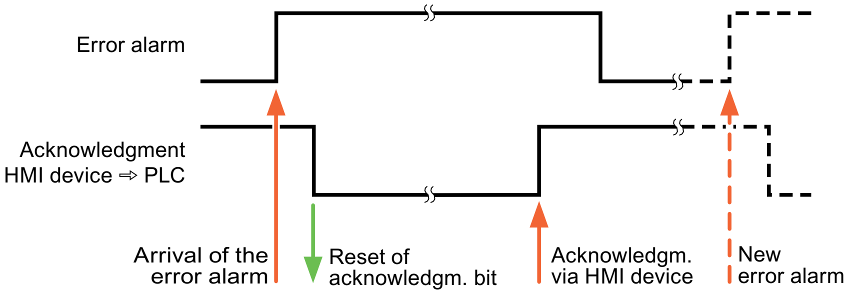
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



Mitsubishi

Mitsubishi communication drivers

Introduction

This section describes the communication between an HMI device and PLCs that use Mitsubishi communication drivers.

The following communication drivers are supported:

- Mitsubishi MC TCP/IP
- Mitsubishi FX

Data exchange

Data is exchanged by means of tags or area pointers.

- Tags

  The PLC and the HMI device use process values for data exchange. You create tags in the configuration that point to addresses in the PLC. The HMI device reads the value from the defined address, and then displays it. The operator may also enter values on the HMI device, which are then written to the address in the PLC.

- Area pointers

  Area pointers are used to exchange specific data and are only set up when these data are used.

Mitsubishi MC TCP/IP

Configuring a connection via Mitsubishi MC TCP/IP

Introduction

You configure a connection to a PLC with a Mitsubishi MC TCPI/IP communication driver in the "Connections" editor of the HMI device.

The Ethernet interfaces are named differently depending on the HMI device.

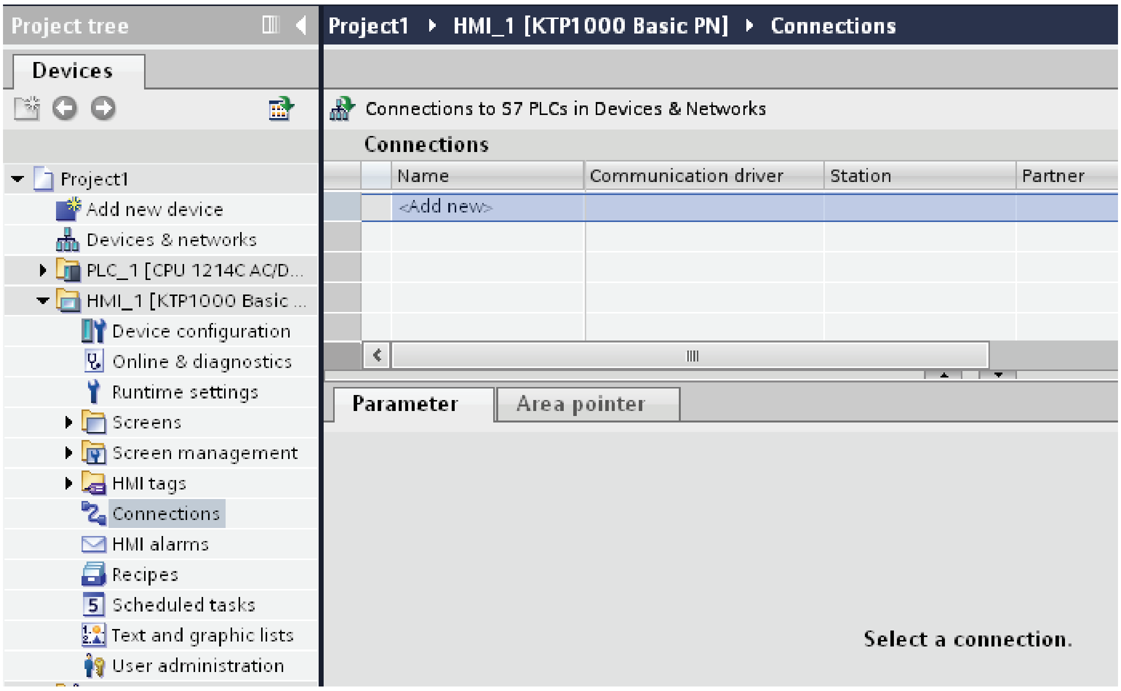
Example: PROFINET interface corresponds to the Ethernet interface

Requirements

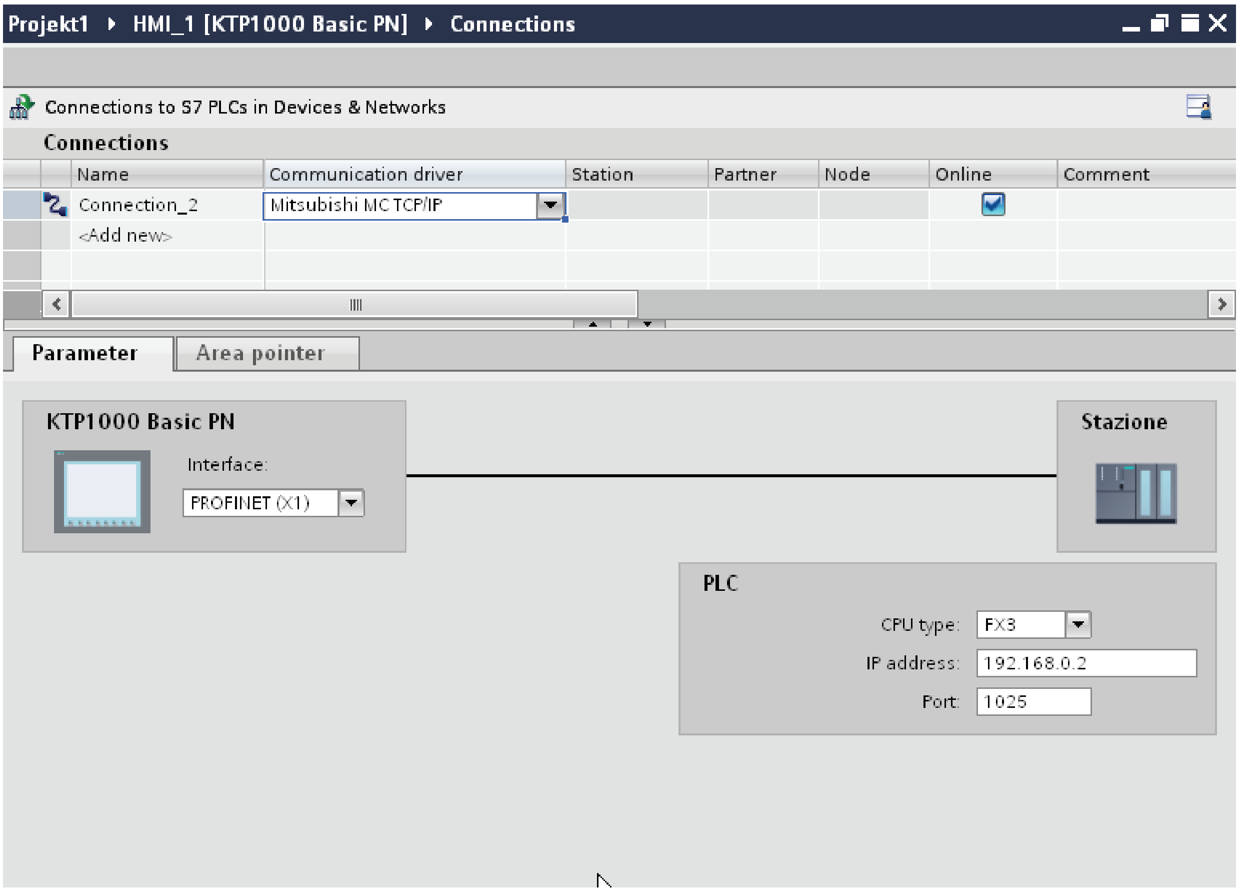
- A project is open.
- An HMI device has been created.

Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.
3. Double-click "<Add>" in the "Connections" editor.



4. In the "Communication drivers" column, select the "Mitsubishi MC TCPI/IP" driver.



Projekt1 ▸ HMI_1 [KTP1000 Basic PN] ▸ Connections

Connections to S7 PLCs in Devices & Networks

Connections

Name	Communication driver	Station	Partner	Node	Online	Comment
Connection_2	Mitsubishi MC TCP/IP				☑	
<Add new>						

KTP1000 Basic PN

Interface: PROFINET (X1)

Stazione

PLC

CPU type: FX3

IP address: 192.168.0.2

Port: 1025

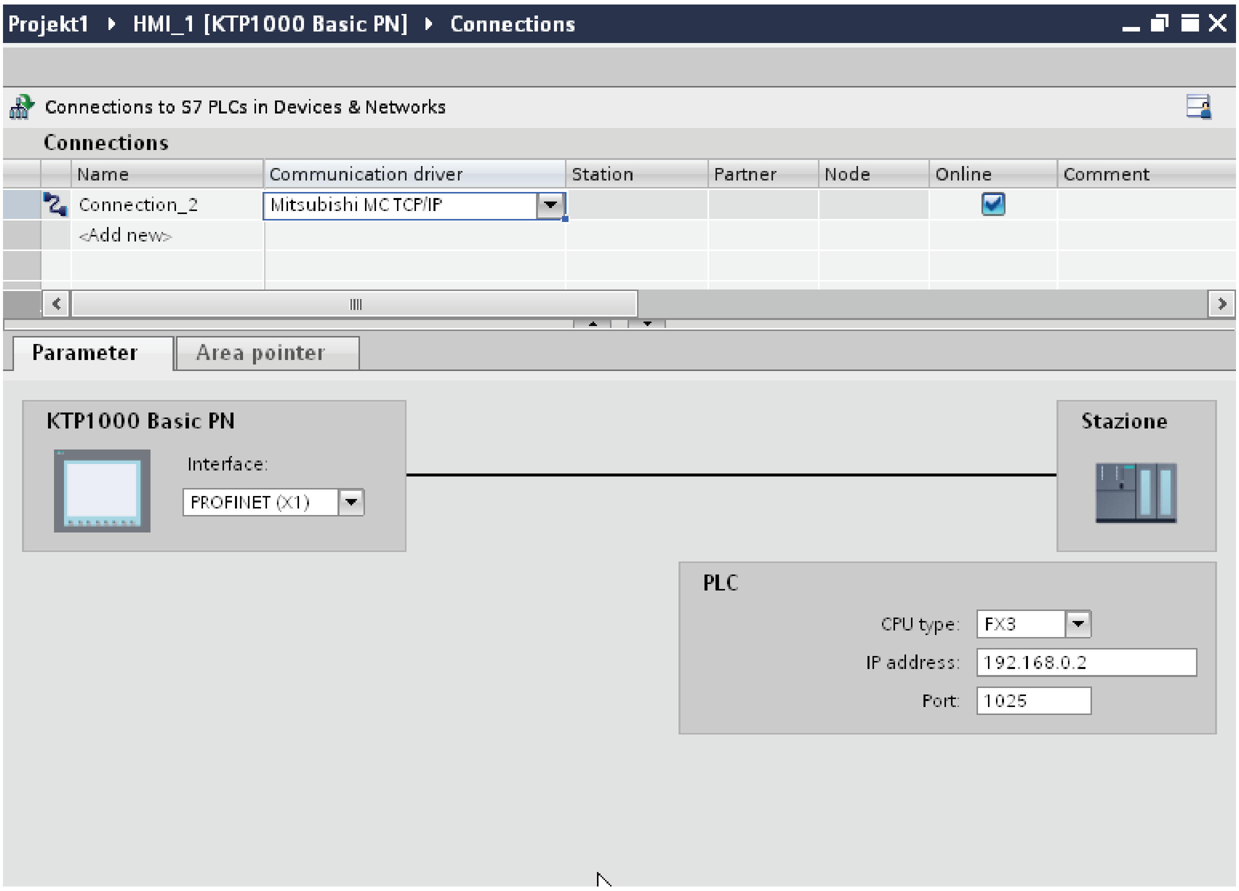
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Mitsubishi MC TCP/IP)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device" and "PLC" areas are available for assigning parameters according to the interface used.



Projekt1 ▸ HMI_1 [KTP1000 Basic PN] ▸ Connections

Connections to S7 PLCs in Devices & Networks

Connections

Name	Communication driver	Station	Partner	Node	Online	Comment
Connection_2	Mitsubishi MC TCP/IP				☑	
<Add new>						

KTP1000 Basic PN

Interface: PROFINET (X1)

Stazione

PLC

CPU type: FX3

IP address: 192.168.0.2

Port: 1025

Parameters for the HMI device

You can select only one interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC. The IP address is transferred to the HMI device during project transfer.

Note

The IP address in the control panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

The IP address already set up in the control panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

To set up the IP address of the HMI device:

1. Click on the HMI device.
2. Open the "Device configuration" editor.
3. Click the Ethernet interface.
4. Assign the IP address in the inspector window under:

   "General > PROFINET interface > Ethernet addresses"

Parameters for the PLC

- CPU type

  For "CPU type", you set the type of PLC to which the HMI device is connected.

  The following settings are possible:

  –FX3

  –Q

  If you select the FX3 CPU type, the Mitsubishi MC protocol "1E" is used and "3E" for the "Q" CPU type.

  The "Binary code" protocol variant is always used.

NOTICE

If the CPU type is changed for a configured connection, tags with the following properties must be revised:

- Operands that do not exist for the new CPU type, such as "W", "B", "F".
- Inputs and outputs with different addressing (hexadecimal/octal)
- Addresses greater than the valid address area of the new CPU type

- IP address

  Set the IP address or host name of the Ethernet/IP module of the PLC. Only the IP address can be used on a Basic Panel.
- Port

  Set the port number of the module of the PLC.

Connecting HMI device to PLC

Connections via Mitsubishi MC TCP/IP

Connection

The HMI device can be connected to the Mitsubishi PLC using the following components:

- Existing Ethernet network that also contains the PLCs
- Cross-over Ethernet cable connected directly to the Ethernet interface of the CPU or the communication module

The connection of the HMI device to a Mitsubishi PLC is limited primarily to the physical connection of the HMI device. Special blocks for the connection are not required in the PLC.

Connect the HMI device to one or several Q-series and/or FX3 PLCs. Connect the HMI device via the following interfaces:

- Communication interface OnBoard
- Approved communication module suitable for the PLC

Note

Timeout response with TCP/IP (Ethernet)

Due to the use of the TCP/IP protocol, the breakdown of a connection is detected at the earliest after approximately one minute. Communication failure cannot be reliably detected if no tags are requested, for example, no output tags in the current screen.

Configure area pointer coordination for each PLC. This setting ensures that a communication failure is recognized after approximately two minutes, even in the aforementioned scenario.

Communication types

Approved communication types

- Only applies for Mitsubishi FX(PG protocol):

  The point-to-point connection from a HMI device to an approved Mitsubishi FX–CPU via Mitsubishi FX is system-tested and approved by Siemens AG.

- Only applies for Mitsubishi MC TCP/IP:

  The following communication types are system-tested and approved:

  – Point-to-point connection to the approved PLCs

  – Multipoint connection from a HMI device with up to 4 PLCs with the respectively approved PLCs. CPU types (FX3 and Q) can be mixed.

Note

The HMI device is a client and the PLC must operate as a server.

Connectable PLCs

Connections can be implemented for the following Mitsubishi PLCs:

PLC	Mitsubishi FX (PG protocol)	Mitsubishi MC TCP/IP
MELSEC FX1n, FX2n	Yes	No
MELSEC FX3U, FX3UC, FX3G with communication module FX3U-ENET	No	Yes
MELSEC System Q • Q-series with the communication module QJ71E71-100 • QnUDEH CPU with Ethernet interface onboard	No	Yes

Parameterization of the communications modules

FX3 PLCs

Procedure

1. Start the FX-Configurator.
2. Select the module.
3. Assign the following settings in the "Operational settings" dialog.
   - Communication data code:
     Binary code
   - Initial timing:
     Always wait for OPEN
   - IP address:
     IP address
   - Send frame setting:
     Ethernet(V2.0)
   - TCP Existence confirmation setting:
     Use the Ping
4. Assign the following settings in the "Open Settings" dialog:
   - Protocol:
     TCP
   - Open system:
     Unpassive
   - Fixed buffer:
     Receive
   - Fixed buffer communication procedure:
     Procedure exist(MC)
   - Pairing open
     Disable
   - Existence confirmation
     No confirm
   - Host station Port No. (DEC)
     Port number

Note

The port number chosen in the communication module must match the port number in WinCC. A connection with a port number must be assigned for each connected HMI device.

5. Confirm the default settings of the other dialog boxes.

The network no. and station no. parameters are not relevant for the connection and can be chosen as required.

Q PLCs

Procedure

1. Click "Edit network parameters".
2. Select the network type:
   – Ethernet

     The network number and the group / station number are not evaluated and can be freely assigned
3. Assign the following settings in the "Operational settings" dialog.
   – Communication data code:

     Binary code

   – Initial timing:

     Always wait for OPEN

   – IP address:

     IP address

   – Send frame setting:

     Ethernet(V2.0)

   – Enable write operations during RUN

4. Assign the following settings in the "Open settings" dialog.
   - Protocol:

     TCP
   - Open system:

     Unpassive
   - Pairing open

     Disable
   - Existence confirmation

     No confirm
   - Host station Port No. (HEX)

     Port-Nummer

   Note

   The port number chosen in the communication module must match the port number in WinCC. A connection with a port number must be assigned for each connected HMI device.

Internal Ethernet port of the Q0xUDEH CPU

Procedure

1. Assign the following settings in the "Internal Ethernet Port" dialog.
   - IP address:

     IP address
   - Communication data code:

     Binary code
   - Enable online changes
2. Assign the following settings in the "Open settings" dialog.
   - Protocol:

     TCP
   - Open system:

     MC-Protocol
   - Host station Port No. (HEX)

     Port number

Note

The port number chosen in the communication module must match the port number in WinCC. A connection with a port number must be assigned for each connected HMI device.

Performance features of communication

Permitted data types for Mitsubishi MC TCPI/IP

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Operand type	Length
4-bit block	M, X, Y, B, F	1 byte
8-bit block	M, X, Y B, F	1 byte
12-bit block	M, X, Y B, F	2 bytes
16-bit block	M, X, Y B, F	2 bytes
20-bit block	M, X, Y B, F	4 bytes
24-bit block	M, X, Y B, F	4 bytes
28-bit block	M, X, Y B, F	4 bytes
32-bit block	M, X, Y B, F	4 bytes
Bool	M, D, X, Y B, F	1-bit
DInt	D, W	4 bytes
DWord	D, C, W	4 bytes
Int	D, W	2 bytes
Real 1)	D, W	4 bytes
String 1)	D, W	1 to 80 characters
Word	D, T, C, W	2 bytes

1) The "String" and "Real" data types are not available for all CPUs.

2) Operand types B, F and W are only available for CPU type "Q".

Note

Note the following for write accesses:

Tags can only be written if "Enable online changes" or "Enable write operations during RUN" was selected when parameterizing the Mitsubishi communication modules.

For data type "Bool" in operand type "D", the entire word is written back to the PLC following a change to the specified bit. There is no check to determine whether any other bits in the word have changed. As a result, the PLC only has read access to the specified word.

Note

Array elements in I/O fields cannot be used in communication with a Mitsubishi PLC.

Supported CPU types for Mitsubishi MC TCP/IP

CPU types

The following CPU types are supported for configuring the Mitsubishi MC TCP/IP communication driver.

- FX3 series
  - FX 3G / FX 3G with communication modul FX3U-ENET
  - FX 3U / FX 3U with communication modul FX3U-ENET
  - FX 3UC / FX 3UC with communication modul FX3U-ENET
- Q series
  - Q-Series with QJ71E71-100 communication module
- iQ series / QnUD
  - QnUDEHCPU with built in ethernet module

Addresses for Mitsubishi MC TCP/IP

Address areas for connections via Mitsubishi MC TCP/IP

The address area boundaries differ for the different series; refer to the Mitsubishi Computerlink manuals for this information.

Examples of address area boundaries dependent on the CPU and communication format:

Name	Operand type	Max. address FX3	Max. address Q-Series
Output/Input	Y/X	Octal X/Y 0 - 267	HEX X/Y 0 - 7FF
Bit memory	M	M0 - M3071 and M8000 - M8255	M/L/S 0 - 8191
Data register	D	D0 - 7999 D8000 - D8255	D0 - 8191 D9000 - D9255 becomes SD1000 - SD1255
Counter	C	C0 - 255	C0 - 1023
Timer	T	T0 - 255	T0 - 2047
Link register	W	--	Hex: W0 - FFF
Link flag	B	--	Hex: B0 - FFF
Error flag	F	--	F0 - 2047

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.

   The project is compiled automatically.

   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Mitsubishi FX

Configuring a connection via Mitsubishi FX

Introduction

You configure a connection to a PLC with a Mitsubishi FX communication driver in the "Connections" editor of the HMI device.

The Mitsubishi FX protocol is also referred to as the Mitsubishi PG protocol.

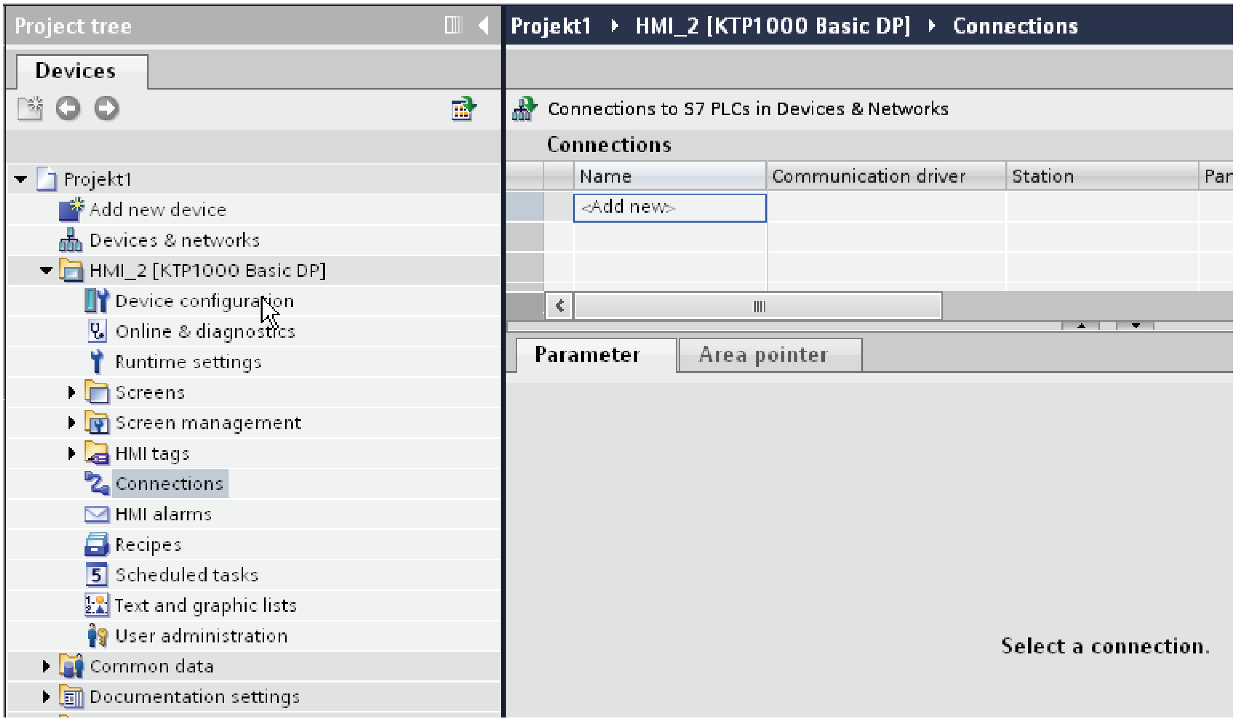
The interfaces are named differently depending on the HMI device.

Requirements

- A project is open.
- An HMI device has been created.

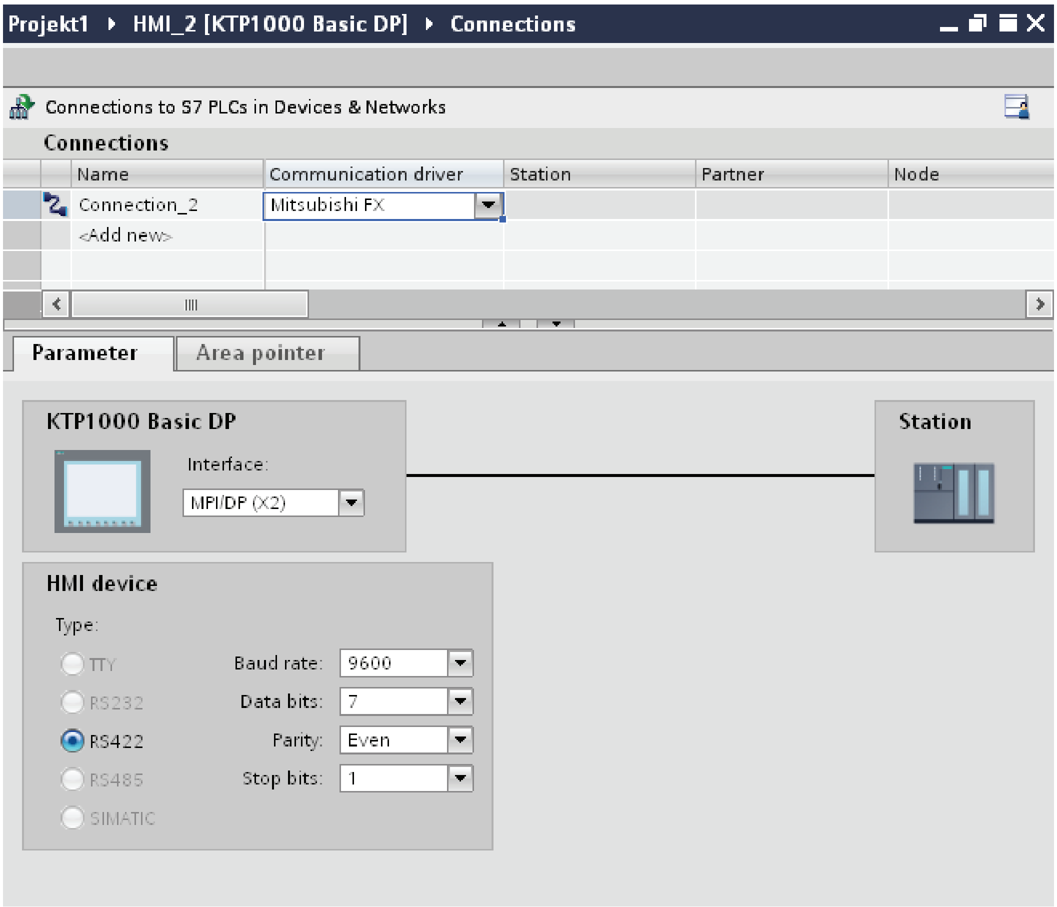
Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.
3. Double-click "<Add>" in the "Connections" editor.



Select a connection.

4. In the "Communication drivers" column, select the "Mitsubishi FX" driver.



Projekt1 ▸ HMI_2 [KTP1000 Basic DP] ▸ Connections

Connections to S7 PLCs in Devices & Networks

Connections

Name	Communication driver	Station	Partner	Node
Connection_2	Mitsubishi FX			
<Add new>				

Parameter | Area pointer

KTP1000 Basic DP

Interface: MPI/DP (X2)

Station

HMI device

Type:

○ TTY
○ RS232
● RS422
○ RS485
○ SIMATIC

Baud rate: 9600
Data bits: 7
Parity: Even
Stop bits: 1

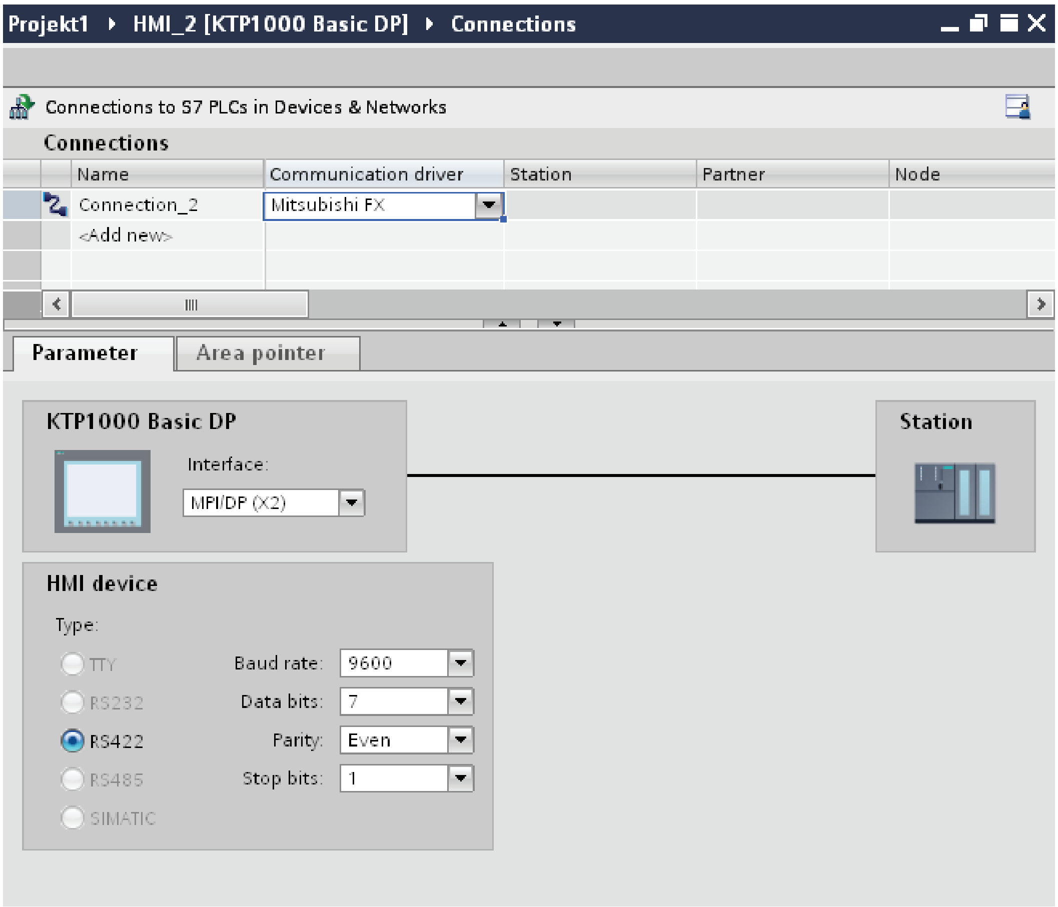
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Mitsubishi FX)

Parameters to be set

To assign the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device", "Network", and "PLC" areas are available for assigning parameters according to the interface used.



Projekt1 ▸ HMI_2 [KTP1000 Basic DP] ▸ Connections
Connections to S7 PLCs in Devices & Networks
Parameter
Area pointer
KTP1000 Basic DP
Station
HMI device

Connections

Name	Communication driver	Station	Partner	Node
Connection_2	Mitsubishi FX			
<Add new>				

Interface: MPI/DP (X2)

Type:

- ○ TTY
- ○ RS232
- ● RS422
- ○ RS485
- ○ SIMATIC

Baud rate: 9600

Data bits: 7

Parity: Even

Stop bits: 1

Parameters for the HMI device

You can select an interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

- "Type"

  Specifies the physical connection used.

  ### Note

  If you use the IF1B interface, you must switch over the RS422 receive data and the RTS signal additionally by 4 DIL-switches on the back of the HMI device.

Parameters for the PLC

- Baud rate

  For "Baud rate", select the transmission speed between the HMI device and PLC.

  Select the baud rate "9600".

- Data bits

  For "Data bits", you can choose between "7 bits" and "8 bits".

  Select "7 bits".

- Parity

  For "Parity", you can choose from "None", "Even", and "Odd".

  Select "Even".

- Stop bits

  For "Stop bits", you can choose between 1 and 2 bits.

  Select "1 bit".

Connecting HMI device to PLC

Communication types

Approved communication types

- Only applies for Mitsubishi FX(PG protocol):

  The point-to-point connection from a HMI device to an approved Mitsubishi FX–CPU via Mitsubishi FX (PG protocol := protocol for access to the program and memory elements of the FX series PC CPU version V1.21 and after) is system-tested and approved by Siemens AG.

- Only applies for Mitsubishi MC TCP/IP:

  The following communication types are system-tested and approved:

  - Point-to-point connection to the approved PLCs
  - Multipoint connection from a HMI device with up to 4 PLCs with the respectively approved PLCs. CPU types (FX3 and Q) can be mixed.

  Note

  The HMI device is a client and the PLC must operate as a server.

Connectable PLCs

Connections can be implemented for the following Mitsubishi PLCs:

PLC	Mitsubishi FX (PG protocol)	Mitsubishi MC TCP/IP
MELSEC FX1n, FX2n	Yes	No
MELSEC FX3U, FX3UC, FX3G with communication module FX3U-ENET	No	Yes
MELSEC System Q • Q-series with the communication module QJ71E71-100 • QnUDEH CPU with Ethernet interface onboard	No	Yes

Connections via Mitsubishi FX

Connection

Connect the HMI device to the programming interface of the CPU (RS 422) (see documentation of the PLC).

The connection between the HMI device and the Mitsubishi PLC is basically restricted to setting the interface parameters. Special blocks for the connection are not required in the PLC.

Connecting cable

The following connecting cables are available to connect the HMI device to the PLC.

Interface to HMI device or adapter	Mitsubishi Electric PLC via FX protocol
	FX1n, Fx2n, Mini DIN, 8–pin
RS 232, 9-pin	Mitsubishi SC-09 [1]
RS 422, 9-pin	Connecting cable RS422-2P

[1] Since the Mitsubishi PLCs communicate via RS 422 as a standard, the Mitsubishi programming cable SC–09 with integrated RS 422/RS 232 adaptor is necessary for connecting a HMI device via RS 232.

Note

Applies only to RS 232:

Cable length is restricted to 0.32 m.

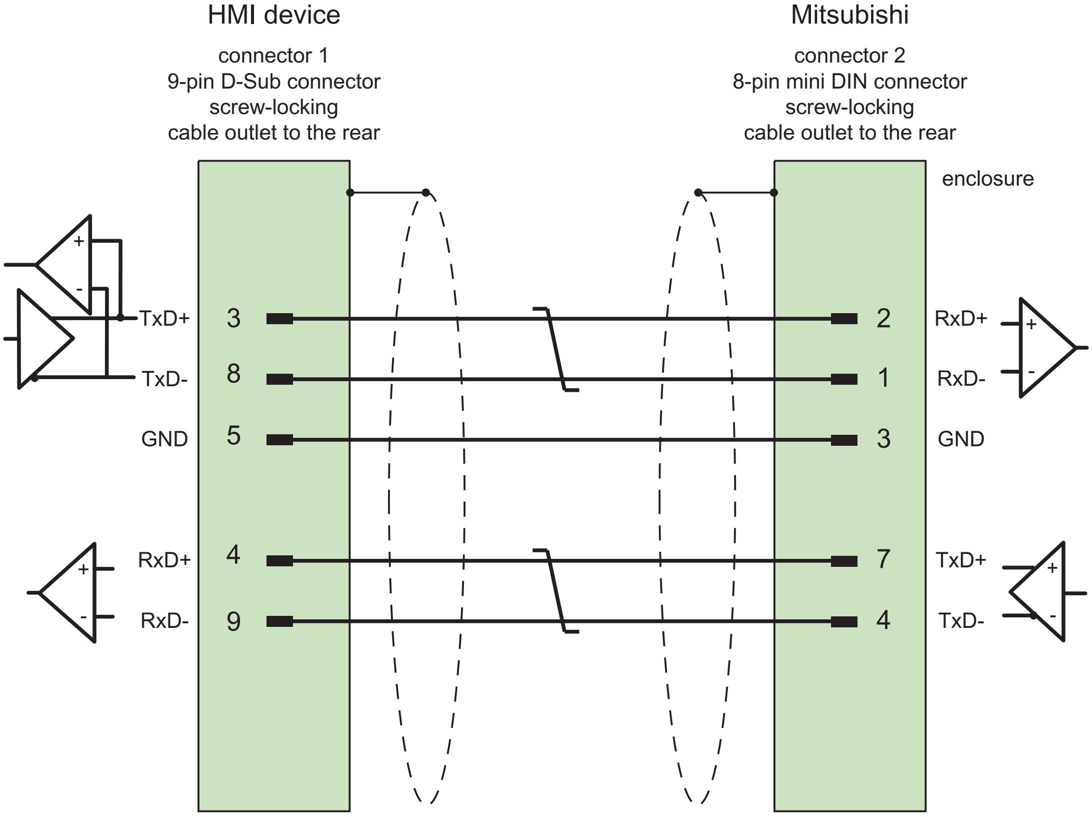
Refer to the relevant device manual to determine which HMI device interface is to be used.

The cable pin assignments can be found in Section "Connecting cables for Mitsubishi FX".

Connecting cables for Mitsubishi FX

Connecting cable RS 422 2P, for Mitsubishi

Connecting cable RS422-2P



Shield with large-area contact to housing at both ends
Cable: 3 x 2 x 0.14 mm^2, shielded,
max. length 500 m

Performance features of communication

Permitted data types for Mitsubishi FX

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Operand type	Length
4-bit block	M, X, Y	1 byte
8-bit block	M, X, Y	1 byte
12-bit block	M, X, Y	2 bytes
16-bit block	M, X, Y	2 bytes
20-bit block	M, X, Y	4 bytes
24-bit block	M, X, Y	4 bytes
28-bit block	M, X, Y	4 bytes
32-bit block	M, X, Y	4 bytes
Bool	D, M, X, Y	1-bit
DWord	D, C-32 bit	4 bytes
Real	D	4 bytes
String	D	1 to 50 characters
Word	D, T, C-16 bit	2 bytes

Note

Note the following for write accesses:

For data type "Bool" in operand type "D", the entire word is written back to the PLC following a change to the specified bit. There is no check to determine whether any other bits in the word have changed. As a result, the PLC only has read access to the specified word.

Note

Array elements in I/O fields cannot be used in communication with a Mitsubishi PLC.

Supported CPU types for Mitsubishi FX

CPU types

The following CPU types are supported for configuring the Mitsubishi FX communication driver.

- FX1 series
  - FX1n
- FX2 series
  - FX2n

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.

   The project is compiled automatically.

   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Data exchange

Area pointers for Mitsubishi

Area pointers for connections via Mitsubishi communication drivers

You use an area pointer to access a data area in the PLC.

For more detailed information on area pointers and their configuration, refer to Section "Data exchange using area pointers".

Special considerations for connections via Mitsubishi communication drivers

You can configure the following area pointers

Area pointer	Mitsubishi MC TCP/IP	Mitsubishi FX
Screen number	Yes	Yes
Date/time	Yes	Yes
Date/time PLC	Yes	Yes
Coordination	Yes	Yes
Project ID	Yes	Yes
Job mailbox	Yes	Yes
Data record	Yes	Yes

Restrictions Mitsubishi MC TCP/IP

The following restrictions apply for configuring area pointers.

CPU type	Data types	Operand type
FX3	Int, Word	D
Q	Int, Word	D

Mitsubishi FX restrictions

You cannot use the D operand type for configuring area pointers.

See also

Data exchange using area pointers (Page 2676)

Trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out time-triggered for Basic Panels.

For additional information see:

Configuring trend displays for values from the PLC (Page 2155)

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration.

Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Alarms

Configuring alarms

Configuring alarms for non-integrated connections

Several steps are necessary to configure alarms such as warnings, error messages and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with alarms (Page 2167)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a Mitsubishi communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
FX1n, FX2n, FX3 series, Q-Series, iQ-Series	Word, Int [1]	4 bit-block, 8 bit-block, 12 bit-block, 16 bit-block, 20 bit-block, 24 bit-block, 28 bit-block, 32 bit-block, Word, DWord, Int [1], DInt [1], Real,
[1] Not for Mitsubishi FX communication driver		

How the bit positions are counted

For connections with a Mitsubishi communication driver, the following counting method applies:

How the bit positions are counted	Left byte							Right byte							
In Mitsubishi PLCs	15							8	7							0
In WinCC you configure:	15							8	7							0

Restrictions on alarms

- Mitsubishi MC TCP/IP

  Only tags of operand type "D" and data types "Word" and "Int" are permitted as trigger tags for discrete alarms. You can use array tags (operand type: "D"; data types: "ARRAY [x..y] of Word" or "ARRAY [x..y] of Int") for discrete alarms.

- Mitsubishi FX

  Only tags of operand type "D" and data type "Word" are permitted as trigger tags for discrete alarms. You can use array tags (operand type "D"; data type "ARRAY [x..y] of Word") for discrete alarms."

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

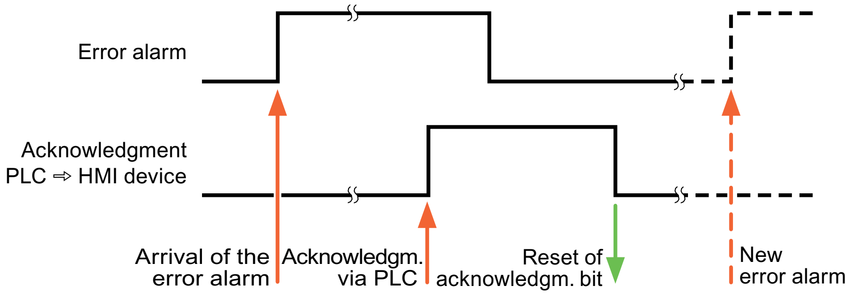
Acknowledgment by the PLC

In "Write acknowledgment tag", you configure the tag or the array tag and the bit number based on which the HMI device can recognize an acknowledgment by the PLC.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In "Read acknowledgment tag", you configure the tag or the array tag and the bit number that is written to the PLC after acknowledgment from the HMI device. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

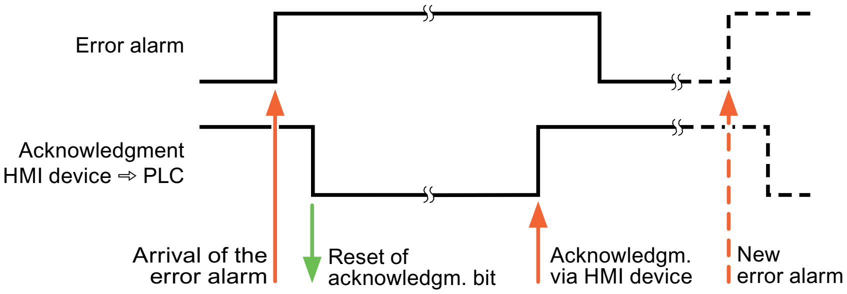
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.


Error alarm
Acknowledgment
HMI device ⇨ PLC
Arrival of the error alarm
Reset of acknowledgm. bit
Acknowledgm. via HMI device
New error alarm

Modicon Modbus

Modicon Modbus communication drivers

Introduction

This section describes the communication between an HMI device and PLCs that use Modicon Modbus communication drivers.

The following communication drivers are supported:

- Modicon Modbus TCP/IP
- Modicon Modbus RTU

Data exchange

Data is exchanged by means of tags or area pointers.

- Tags

  The PLC and the HMI device use process values for data exchange. You create tags in the configuration that point to addresses in the PLC. The HMI device reads the value from the defined address, and then displays it. The operator may also enter values on the HMI device, which are then written to the address in the PLC.

- Area pointers

  Area pointers are used to exchange specific data and are only set up when these data are used.

Modicon Modbus TCP/IP

Configuring a connection via Modicon Modbus TCP/IP

Introduction

You configure a connection to one of the PLCs with Modicon Modbus TCP/IP communication driver in the "Connections" editor of the HMI device.

The Ethernet interfaces are named differently depending on the HMI device.

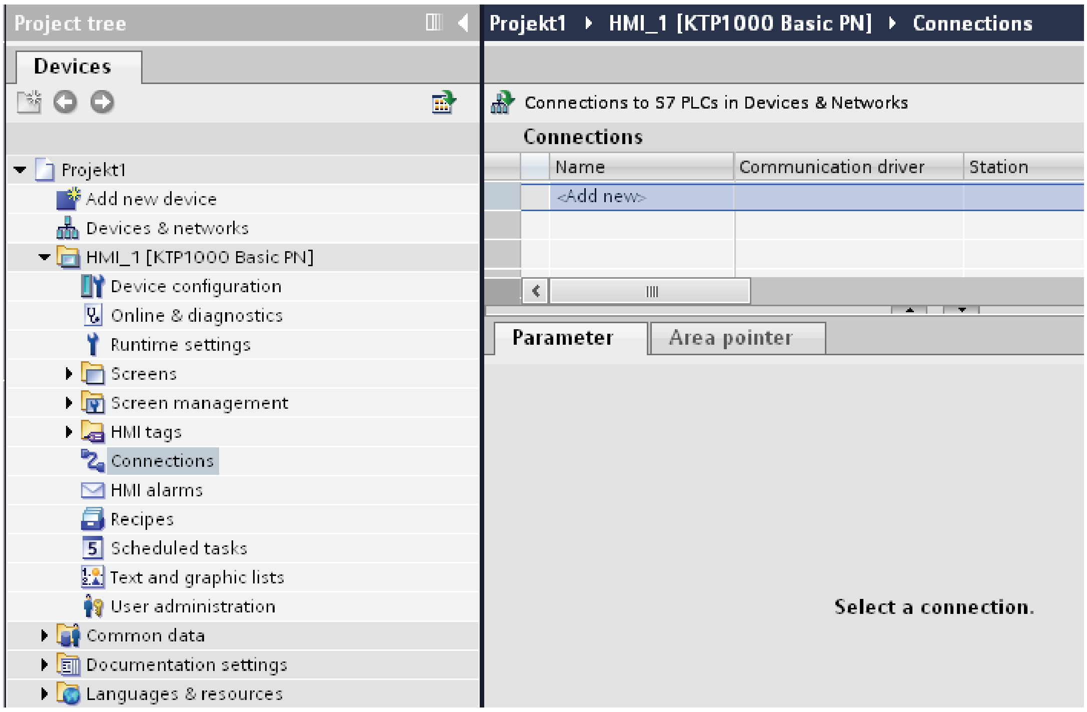
Example: PROFINET interface corresponds to the Ethernet interface

Requirements

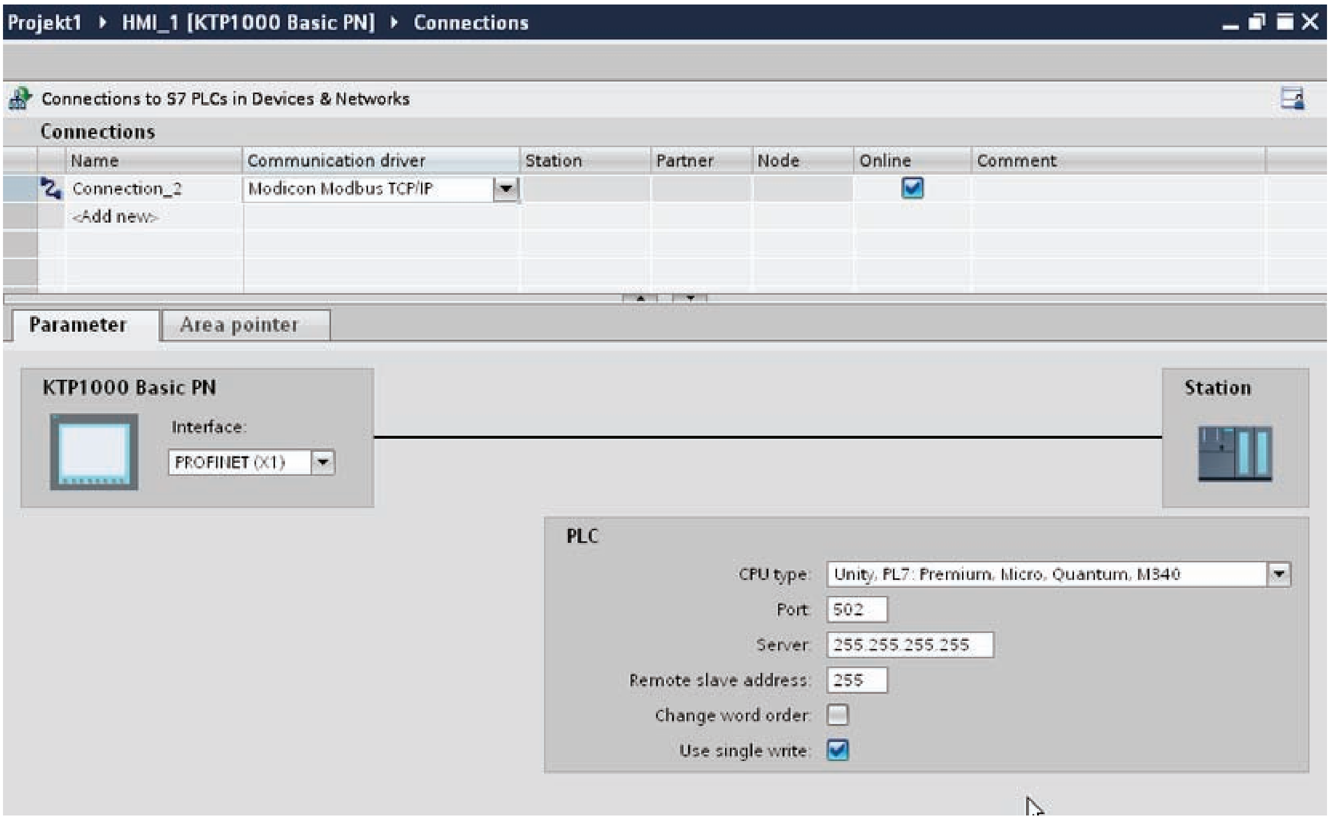
- A project is open.
- An HMI device has been created.

Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.
3. Double-click "<Add>" in the "Connections" editor.



4. Select the "Modicon Modbus TCP" driver in the "Communication driver" column.



Projekt1 ▸ HMI_1 [KTP1000 Basic PN] ▸ Connections
Connections to S7 PLCs in Devices & Networks

Connections

Name	Communication driver	Station	Partner	Node	Online	Comment
Connection_2	Modicon Modbus TCP/IP				☑	
<Add new>						

Parameter
Area pointer
KTP1000 Basic PN
Station

Interface: PROFINET (X1)

PLC

CPU type: Unity, PL7: Premium, Micro, Quantum, M340

Port: 502

Server: 255.255.255.255

Remote slave address: 255

Change word order: ☐

Use single write: ☑

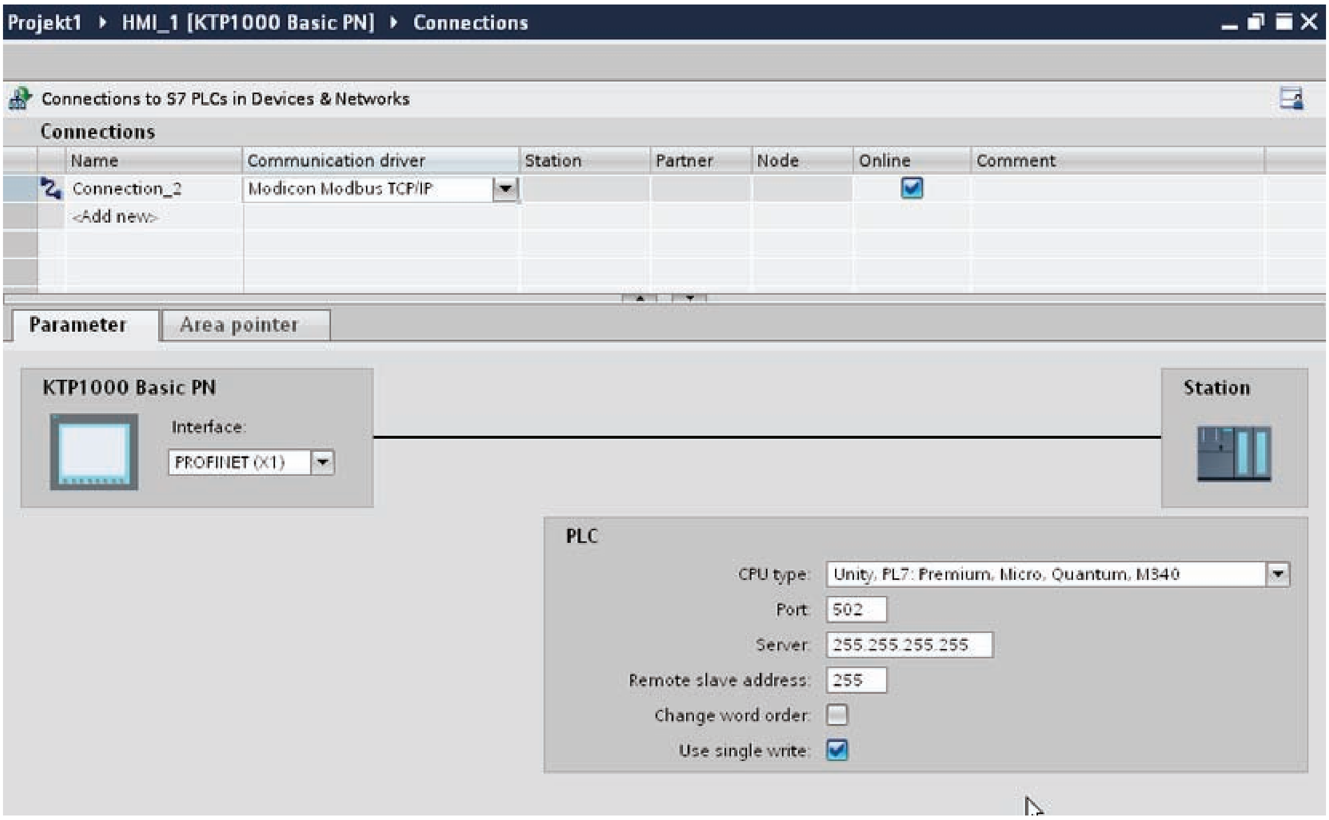
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Modicon Modbus TCP/IP)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device" and "PLC" areas are available for assigning parameters according to the interface used.


Projekt1 ▸ HMI_1 [KTP1000 Basic PN] ▸ Connections
Connections to S7 PLCs in Devices & Networks
Connections
Parameter
Area pointer
Station

Name	Communication driver	Station	Partner	Node	Online	Comment
Connection_2	Modicon Modbus TCP/IP				☑	
<Add new>						

KTP1000 Basic PN

Interface: PROFINET (X1)

PLC

CPU type: Unity, PL7: Premium, Micro, Quantum, M340

Port: 502

Server: 255.255.255.255

Remote slave address: 255

Change word order: □

Use single write: ☑

Parameters for the HMI device

You can select only one interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC. The IP address is transferred to the HMI device during project transfer.

Note

The IP address in the control panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

The IP address already set up in the control panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

To set up the IP address of the HMI device:

1. Click the HMI device.
2. Open the "Device configuration" editor.
3. Click the Ethernet interface.
4. Assign the IP address in the inspector window under:

   "General > PROFINET interface > Ethernet addresses"

Parameters for the PLC

- "CPU type"

  For "CPU type", you set the Modicon PLC to which the HMI device is connected.

- "Port"

  For "Port", you set the port that is used for the TCP/IP connection. The port used by the Modicon PLCs is 502.

- "Server"

  You set the IP address or host name of the PLC under "Server". Only the IP address can be used on a Basic Panel.

- "Remote Slave address"

  Under "Remote Slave address" you only set which slave address the remote PLC has when using a bridge.
  If no bridge is used, the default value 255 (or 0) must be retained.

- "Change word order"

  The "Change word order " parameter only affects the word order of the 32-bit values display. The setting pertains to the data types Double, Double+/-, and Float. The byte order cannot be changed.

  – "Change word order" not activated

    The most significant byte is sent first.

    For double words, the least significant word is sent before the most significant word.

    This setting has been system-tested for all approved PLCs.

  – "Change word order" activated

    The most significant byte is sent first.

    For double words, the most significant word is sent before the least significant word.

    Note

    This setting must be used for the SIEMENS SENTRON PAC3200 and PAC4200 multi-function meters and can be used for PLCs of other manufacturers.

- "Use single write"

  If you deselect this function, only function codes 15H and 16H are used for writing into the PLC.

  If this function remains selected, the function codes 05H, 06H 16H and 16H are used.

Connecting HMI device to PLC

Connections via Modicon Modbus TCP/IP

Connection

The HMI device can be connected to the Modicon Modbus PLC using the following components:

- Existing Ethernet network that also contains the PLCs
- Cross-over Ethernet cable connected directly to the Ethernet interface of the CPU or the communication module

The connection of the HMI device to a Modicon Modbus PLC is limited primarily to the physical connection of the HMI device. Special blocks for the connection are not required in the PLC.

Note

Timeout response with TCP/IP (Ethernet)

Due to the use of the TCP/IP protocol, the breakdown of a connection is detected at the earliest after approximately one minute. Communication failure cannot be reliably detected if no tags are requested, for example, no output tags in the current screen.

Configure area pointer coordination for each PLC. This setting ensures that a communication failure is recognized after approximately two minutes, even in the aforementioned scenario.

Communication types

Enabled types of communication

The following communication types are system-tested and enabled:

- Point-to-point coupling:
- Multiple point coupling of a HMI device (Modbus TCP/IP Client) with up to 4 PLCs, each with different couplings. It is possible to mix the types of CPU.

  The following couplings are possible:

  - Coupling the Ethernet CPU interface of the TSX Unity Quantum.
  - Coupling via the communication modules for Ethernet 140 NOE 771 01 for the TSX Quantum and TSX Unity Quantum series
  - Coupling via the Ethernet interface of the 171 CCC 980 30 CPU adapter of the Momentum series
  - Coupling the Ethernet CPU interface of the TSX Unity Premium.
  - Coupling via the Ethernet TCP/IP connect module TSX ETY 110 for the TSX Premium and TSX Unity Premium series
  - Coupling via the Ethernet TCP/IP connect module TSX ETY 410 for the Micro series
  - Coupling via the Ethernet TCP/IP Modbus Plus Bridge 174 CEV 200 40 to the Modbus Plus interface of the Compact, the TSX Quantum and the TSX Unity Quantum series

Via the TCP/IP Modbus Plus Bridge, 174 CEV 200 40, the PLCs can be accessed at their Remote Slave Address via the Ethernet interface of this bridge.

Note

Integration of the HMI device in a Modbus network via a bridge is not possible. The HMI device is the Modbus master.

Restrictions

The coupling of the HMI device to PLCs of other manufacturers who offer a Modbus TCP/IP interface is not system-tested and thus, not enabled.

However, if another PLC is to be used, observe the following instructions:

- Use the following CPU types, because these operate without address offset and in the usual bit count manner.
  - Unity, PL7: Premium, Micro, Quantum, M340
- The following function codes are used for the respective data areas:

Reading function codes		Address range	
01	ReadCoilStatus	0x / %M	DIGITAL_OUT
02	ReadInputStatus	1x / %I	DIGITAL_IN
03	ReadHoldingRegisters	4x / %MW	USERDATA
04	ReadInputRegisters	3x / %IW	ANALOG_IN
20 (14Hex)	ReadGeneralReference	6x / –	EXTENDEDMEMORY (not for all CPUs)

Writing function codes		Address range	
06 [1]	PresetSingleRegister	4x / %MW	USERDATA Single
16 (10Hex)	PresetMultipleRegisters	4x / %MW	USERDATA Multiple
05 [1]	ForceSingleCoil	0x / %M	DIGITAL_OUT with BIT
15 (0FHex)	ForceMultipleCoils	0x / %M	DIGITAL_OUT with 16 BIT GROUP
21 (15Hex)	WriteGeneralReference	6x / –	EXTENDEDMEMORY (not for all CPUs)

[1] Select use with "Use single write".

Connectable PLCs

Connections can be implemented for the following Modicon Modbus PLCs:

Modicon Modbus PLC	Supported protocol	
	Modicon Modbus RTU [2]	Modicon Modbus TCP/IP
TSX Compact	x	x [1]
TSX Quantum	x	x
Momentum	x	x
Premium	-	x
Micro	-	x
M340 20X0 (without 2010)	-	x

[1] Only via Ethernet TCP/IP Modbus Plus Bridge

[2] Communication via RS 232 is tested and enabled for the PLC. In the HMIs that only have a RS 422/485 interface, the RS 422/232 converter with the order number 6AV6 671-8XE00-0AX0 was tested and enabled.

Performance features of communication

Permissible data types for Modicon Modbus TCP/IP

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

Note

If you change the Modicon Modbus RTU communication driver to Modicon Modbus TCP/IP, the string in the "String" data type may be different.

Permitted data types for CPU type "Unity, PLC: Premium, Micro, Quantum M340"

Data type	Operand type	Length
+/- Double	%MW	4 bytes
+/- Int	%MW, %IW	2 bytes
16-bit group	%MW, %I	2 bytes
ASCII	%MW	0 to 80 characters
Bit	%MW, %IW, %M, %I	1-bit
Double	%MW	4 bytes
Float	%MW	4 bytes
Int	%MW, %IW	2 bytes

Note

The ranges "%I" and "%IW" are not supported for the following CPU types:

- Premium
- Micro
- M340

Permitted data types for CPU type "Concept, ProWORX: Compact, Quantum, Momentum"

Data type	Operand type	Length
+/- Double	4x, 6x	4 bytes
+/- Int	3x, 4x, 6x	2 bytes
16-bit group	0x, 1x	2 bytes
ASCII	4x, 6x	0 to 80 characters
Bit	0x, 1x, 3x, 4x, 6x	1-bit
Double	4x, 6x	4 bytes
Float	4x, 6x	4 bytes
Int	3x, 4x, 6x	2 bytes

Bit counting method

The usual bit counting method "16 LSB - 1 MSB" in the following CPU types is only used in the "HMI tags" editor with the selected "Bit" data type:

- Concept, ProWORX: Compact, Quantum, Momentum

The following bit location assignment applies:

	Left byte								Right byte							
Counting with tags	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Format for "Signed"

The placeholder "+/-" stands for the data types "Signed Int" and "Signed Double".

Supported CPU types for Modicon Modbus TCP/IP

CPU types

The following CPU types are supported for configuring the Modicon Modbus TCP/IP communication driver.

- Compact
- Momentum
- Quantum
  - Concept Quantum
  - Unity Quantum
- Micro
- Premium
- Modicon M340
  - 20x0 (except 2010)

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.

   The project is compiled automatically.

   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Modicon Modbus RTU

Configuring a connection via Modicon Modbus RTU

Introduction

You configure a connection to a PLC with a Modicon Modbus RTU communication driver in the "Connections" editor of the HMI device.

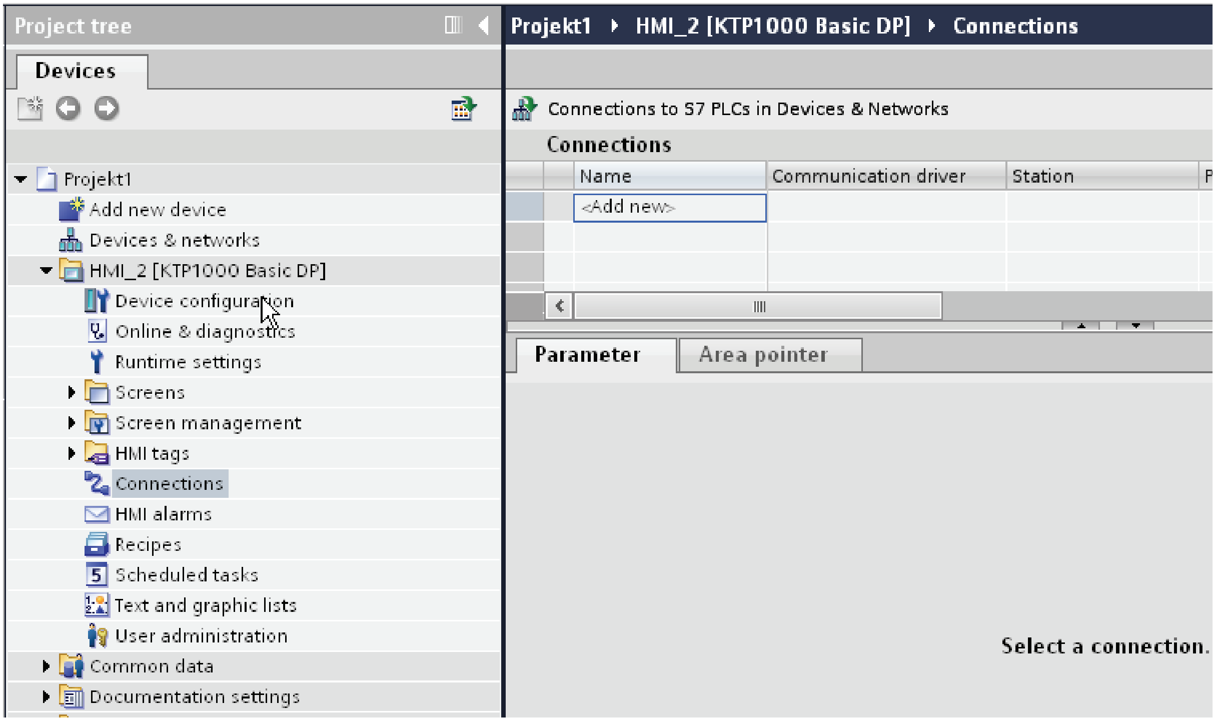
The interfaces are named differently depending on the HMI device.

Requirements

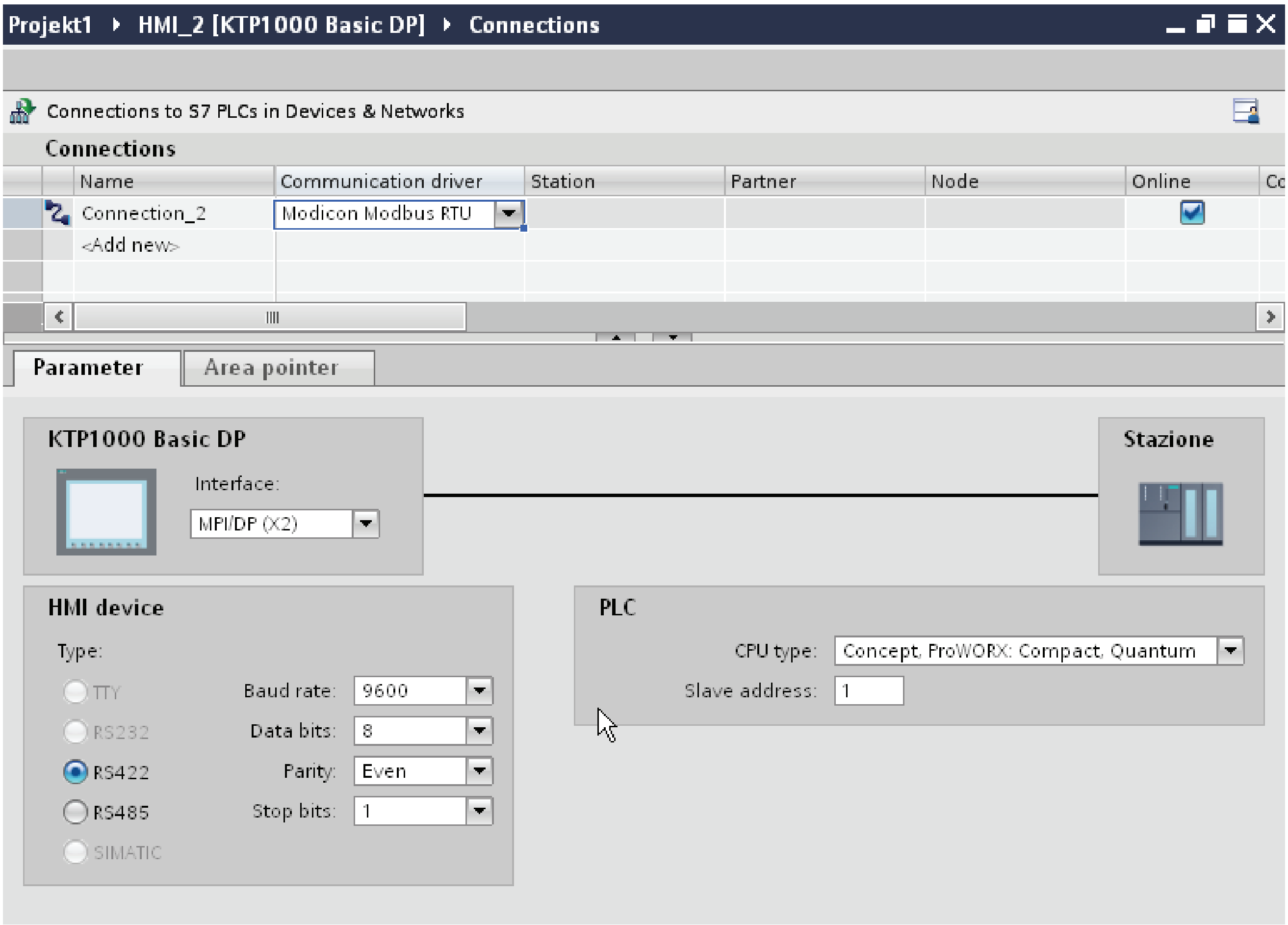
- A project is open.
- An HMI device has been created.

Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.
3. Double-click "<Add>" in the "Connections" editor.



4. In the "Communication drivers" column, select the "Modicon Modbus RTU" driver.



Projekt1 ▸ HMI_2 [KTP1000 Basic DP] ▸ Connections

Connections to S7 PLCs in Devices & Networks

Connections

Name	Communication driver	Station	Partner	Node	Online	Co
Connection_2	Modicon Modbus RTU				☑	
<Add new>						

KTP1000 Basic DP

Interface: MPI/DP (X2)

Stazione

HMI device

Type:

○ TTY
○ RS232
● RS422
○ RS485
○ SIMATIC

Baud rate: 9600
Data bits: 8
Parity: Even
Stop bits: 1

PLC

CPU type: Concept, ProWORX: Compact, Quantum
Slave address: 1

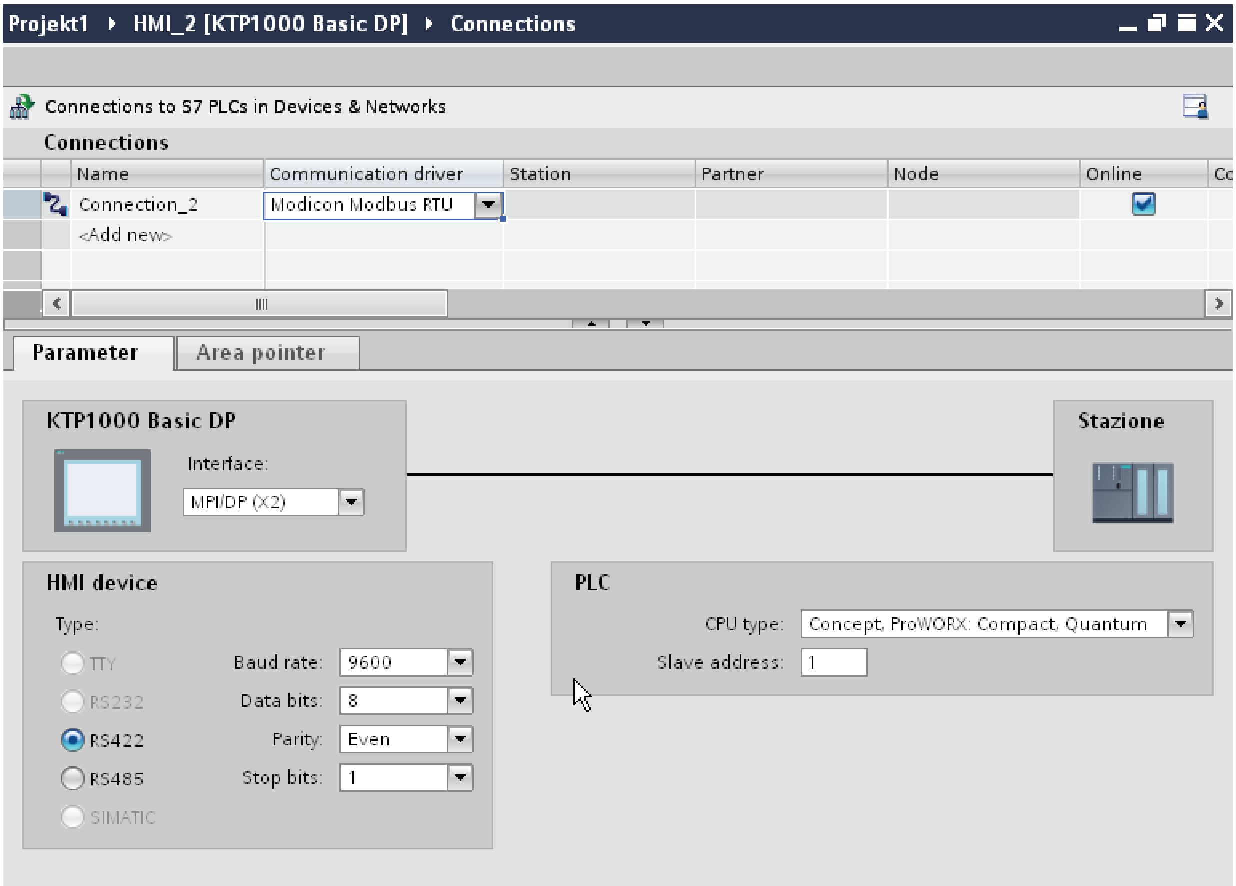
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Modicon Modbus RTU)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device" and "PLC" areas are available for assigning parameters according to the interface used.



Projekt1 ▸ HMI_2 [KTP1000 Basic DP] ▸ Connections
Connections to S7 PLCs in Devices & Networks
Connections
Parameter
Area pointer

Name	Communication driver	Station	Partner	Node	Online	Co
Connection_2	Modicon Modbus RTU				☑	
<Add new>						

KTP1000 Basic DP

Interface: MPI/DP (X2)

Stazione

HMI device

Type:

○ TTY
○ RS232
● RS422
○ RS485
○ SIMATIC

Baud rate: 9600
Data bits: 8
Parity: Even
Stop bits: 1

PLC

CPU type: Concept, ProWORX: Compact, Quantum
Slave address: 1

Parameters for the HMI device

You can select an interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

- Type

  Only RS 232 is system-tested.

  No warranty is given for RS 485.

  Note

  RS 422 is only approved in combination with the RS 422-RS 232 converter.

  Order number: 6AV6 671-8XE00-0AX0

  Note

  If you use the IF1B interface, you must switch over the RS422 receive data additionally by 4 DIL-switches on the back of the HMI device.

- Baud rate

  For "Baud rate", select the transmission speed between the HMI device and Modicon PLC. A baud rate of 19200 or 9600 can be selected for the communication.

  A baud rate of 4800 can be selected for certain HMI devices.

- Data bits

  For "Data bits", only the value "8" can be selected.

- Parity

  For "Parity", you can choose from "None", "Even", and "Odd".

- Stop bits

  For "Stop bits", you can choose between 1 and 2 bits.

Parameters for the PLC

- CPU type

  For "CPU type", you set the Modicon PLC to which the HMI device is connected.

  You can select the following CPUs:

  – Concept, ProWORX: Compact, Quantum

- Slave address

  Under "Slave address" you set which slave address the CPU has.

Connecting HMI device to PLC

Connections via Modicon Modbus RTU

Connection

Connect the HMI device to the Modicon Modbus RTU interface of the Modicon Modbus RTU slave.

The connection of the HMI device to Modicon is limited primarily to the physical connection of the HMI device. Special blocks for the connection are not required in the PLC.

Connection cable

The following connecting cables are available to connect the HMI device to Modicon Modbus.

Interface to the HMI device	Modicon PLC		
	directly via Modbus interface (RS232) 9-pin Sub D male connector	Via MB Bridge (RS 232)	directly via Modbus interface (RS232) 8-pin RJ45 connector
RS 232, 9-pin	PP1	PP1	PP2

The cable pin assignments can be found in Section "Connecting cables for Modicon Modbus RTU".

Communication types

Enabled types of communication

The following communication types are system-tested and enabled:

- Ppoint-to-point connection only via the RS-232 interface.
- Multipoint connection from a HMI device (Modbus-Master) with up to 4 PLCs: The HMI device must be connected with a Modbus Plus Bridge or a Compact, Momentum CPU or TSX Quantum CPU which is configured as a Modbus Plus Bridge.
- You connect the other PLCs via the Modbus Plus connection on the first PLC. The PLCs can be reached under their address via the bridge functionality of the first PLC.

  Note

  It is not possible to integrate the HMI device into a Modbus network because the HMI device is Modbus-Master.

- Integration of the HMI device into a Modbus Plus network via the "bridge mode" of the Compact, Momentum or Quantum (logical point-to-point communication of the HMI device with a Compact, Momentum or Quantum).

Restrictions

The connection of the HMI device to PLCs of other manufacturers which offer a Modicon Modbus interface is not system-tested and therefore not approved.

If you use another PLC nevertheless, observe the following information:

- These drivers only work for tags with the bit counting method typical for Modicon PLCs from left (bit1 = most significant bit) to right (bit16 = least significant bit in data type INT).
- The address offset displayed in the configuring is subtracted at protocol level in the message frame. E.g. in Holding Register 4x the offset "40001". The configured address "40006" therefore becomes address "5" in the message frame. The address (e.g. "5") transferred in the message frame is transformed to the PLC-specific address range in the different Non-Modicon PLCs.
- A reply message frame without "ExceptionCode" is expected within 500 ms.
- The following function codes are used for the respective data areas:

Reading function codes		Address range	
01	ReadCoilStatus	0x	DIGITAL_OUT
02	ReadInputStatus	1x	DIGITAL_IN
03	ReadHoldingRegisters	4x	USERDATA
04	ReadInputRegisters	3x	ANALOG_IN
20 (14Hex)	ReadGeneralReference	6x	EXTENDEDMEMORY (not for all CPUs)

Writing function codes		Address range	
06	PresetSingleRegister	4x	USERDATA Single
16 (10Hex)	PresetMultipleRegisters	4x	USERDATA Multiple
05	ForceSingleCoil	0x	DIGITAL_OUT with data type Bit
15 (0FHex)	ForceMultipleCoils	0x	DIGITAL_OUT with data type 16 bit group
21 (15Hex)	WriteGeneralReference	6x	EXTENDEDMEMORY (not for all CPUs)

Connectable PLCs

Connections can be implemented for the following Modicon Modbus PLCs:

Modicon Modbus PLC	Supported protocol	
	Modicon Modbus RTU [2)]	Modicon Modbus TCP/IP
TSX Compact	x	x [1)]
TSX Quantum	x	x
Momentum	x	x
Premium	-	x

	Supported protocol	
Micro	-	x
M340 20x0 (without 2010)	-	x

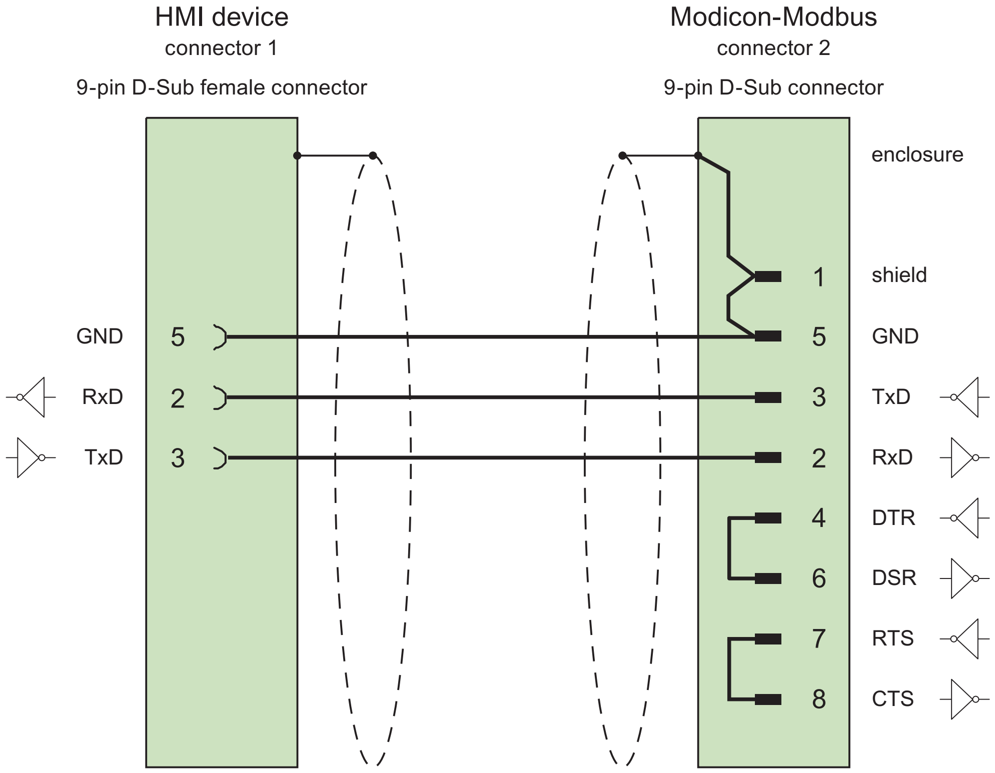
[1] Only via Ethernet TCP/IP-Modbus Plus Bridge

[2] Communication via RS 232 is tested and enabled for the PLC. In the HMI devices which only have an RS 422/485 interface, the RS 422/232 converter with the order number 6AV6 671-8XE00-0AX0 was tested and approved.

Connecting cables for Modicon Modbus RTU

Connecting cable PP1, RS-232, for Modicon

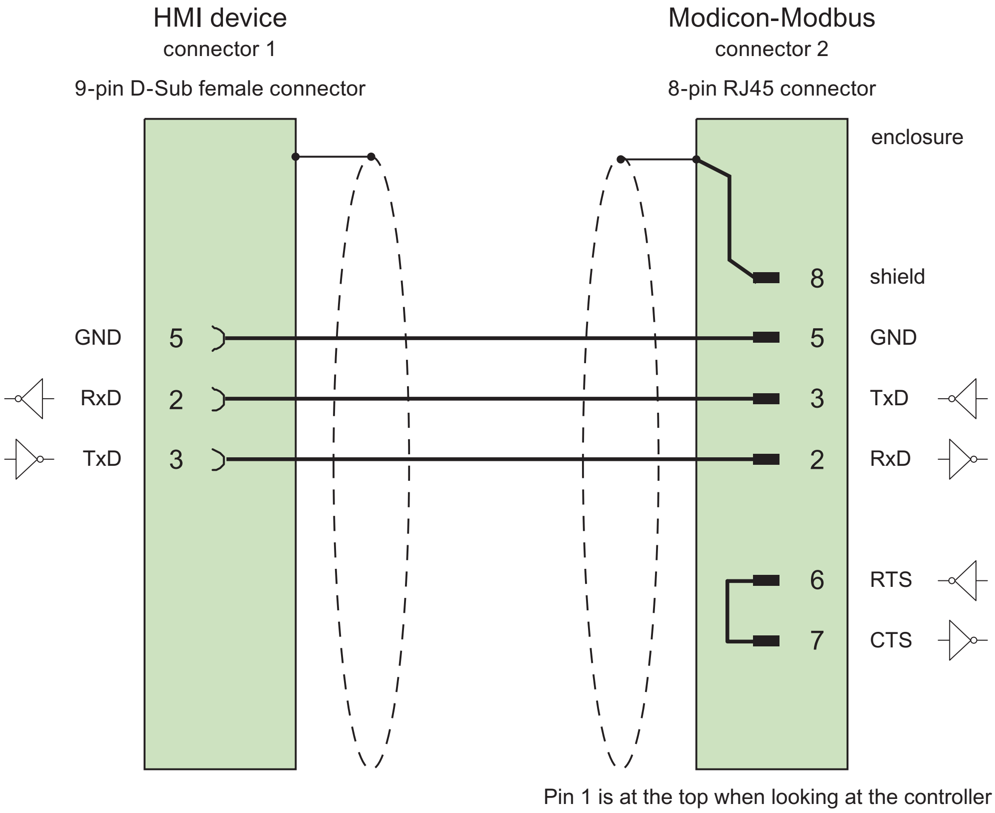
Point-to-point cable 1: PLC > PC ...



Cables: 3 x 0.14 mm², shielded,
max. length 15 m

Connecting cable PP2, RS-232, for Modicon

Point-to-point cable 2: PLC (TSX Compact) > PC...



Cables: 3 x 0.14 mm^2, shielded,
max. length 15 m

Performance features of communication

Permitted data types for Modicon Modbus RTU

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Operand type	Length
+/- Double	4x, 6x	4 bytes
+/- Int	3x, 4x, 6x	2 bytes
16-bit group	0x, 1x	2 bytes
ASCII	4x, 6x	0 to 80 characters
Bit [1]	0x, 1x, 3x, 4x, 6x	1-bit
Double	4x, 6x	4 bytes
Float	4x, 6x	4 bytes
Int	3x, 4x, 6x	2 bytes

[1] Note the following for write accesses:

For data type "Bit" with the operand types "4x" and "6x", the entire word is written back to the PLC following a change to the specified bit. There is no check to determine whether any other bits in the word have changed. As a result, the PLC only has read access to the specified word.

The usual bit counting method (16 LSB - 1 MSB) in the following CPU types is only used in the "HMI tags" editor with the selected "Bit" data type:

- Concept ProWORX: Compact, Quantum

The following bit location assignment applies:

	Left byte								Right byte							
Counting with tags	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Format for "Signed"

The placeholder "+/-" stands for the data types "Signed Int" and "Signed Double".

Supported CPU types for Modicon Modbus RTU

CPU types

The following CPU types are supported in the configuration of the Modicon Modbus RTU communication driver.

- Compact
- Momentum
- Quantum

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.

   The project is compiled automatically.

   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Data exchange

Area pointers for Modicon Modbus

Area pointers for connections via Modicon Modbus communication drivers

You use an area pointer to access a data area in the PLC.

For more detailed information on area pointers and their configuration, refer to Section "Data exchange using area pointers (Page 2676)".

Special considerations for connections via Modicon communication drivers

You can configure the following area pointers

Area pointer	Modicon Modbus TCP/IP	Modicon Modbus RTU
Screen number	Yes	Yes
Date/time	Yes	Yes
Date/time PLC	Yes	Yes
Coordination	Yes	Yes
Project ID	Yes	Yes
Job mailbox	Yes	Yes
Data record	Yes	Yes

Restrictions Modicon Modbus TCP/IP

The following restrictions apply for configuring area pointers.

CPU type	Data types	File types
Concept, ProWORX: Compact, Quantum, Momentum	+/- Int, Int	4x, 6x
Unity, PL7: Premium, Micro, Quantum, M340	+/- Int, Int	%MW

Modicon Modbus RTU restrictions

The following restrictions apply for configuring area pointers.

CPU type	Data types	File types
Concept, ProWORX: Compact, Quantum, Momentum	+/- Int, Int	4x, 6x

Trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out time-triggered for Basic Panels.

For additional information see:

Configuring trend displays for values from the PLC (Page 2155)

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration.

Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Alarms

Configuring alarms

Configuring alarms for non-integrated connections

Several steps are necessary to configure alarms such as warnings, error messages and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with alarms (Page 2167)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a Modicon Modbus communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
All Modicon series	Int, +/-Int	16 Bit Group, Int, +/-Int, Double, +/-Double, Float

Arrays and array tags cannot be used for discrete alarms.

How the bit positions are counted

For connections with a Modicon Modbus communication driver, the following counting method applies:

How the bit positions are counted	Left byte								Right byte							
In WinCC you configure:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

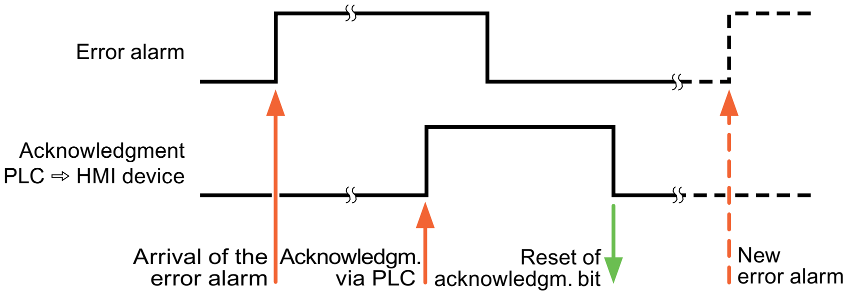
Acknowledgment by the PLC

In "Write acknowledgment tag", you configure the tag or the array tag and the bit number based on which the HMI device can recognize an acknowledgment by the PLC.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In "Read acknowledgment tag", you configure the tag or the array tag and the bit number that is written to the PLC after acknowledgment from the HMI device. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

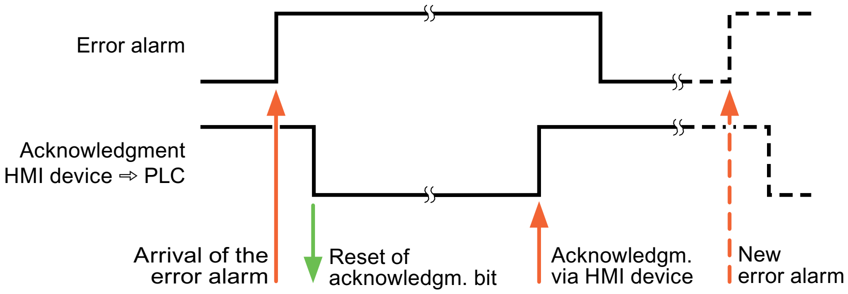
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



Omron

Omron communication drivers

Introduction

This section describes the communication between an HMI device and PLCs that use Omron communication drivers.

The following communication drivers are supported:

- Omron Host Link

Data exchange

Data is exchanged by means of tags or area pointers.

- Tags

  The PLC and the HMI device use process values for data exchange. You create tags in the configuration that point to addresses in the PLC. The HMI device reads the value from the defined address, and then displays it. The operator may also enter values on the HMI device, which are then written to the address in the PLC.

- Area pointers

  Area pointers are used to exchange specific data and are only set up when these data are used.

Omron Host Link

Configuring a connection via Omron Host Link

Introduction

You configure a connection to a PLC with an Omron Host Link communication driver in the "Connections" editor of the HMI device.

Note

Connection with Omron Host Link

A connection will not automatically be established when runtime is started if you have configured a connection via Omron.

A tag which is in the valid PLC memory area must be configured in the runtime start screen.

The connection will otherwise only be established once a corresponding screen has been selected.

This tag will be accessed when runtime is started and a connection will then be established.

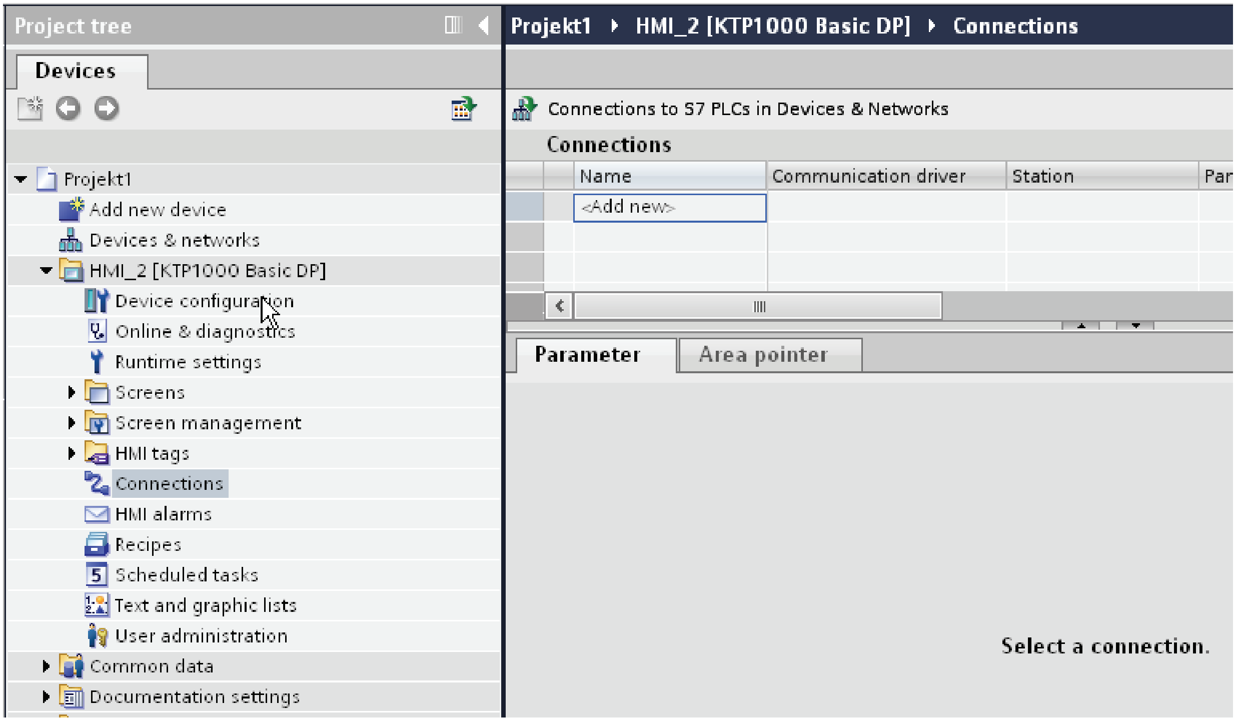
The interfaces are named differently depending on the HMI device.

Requirements

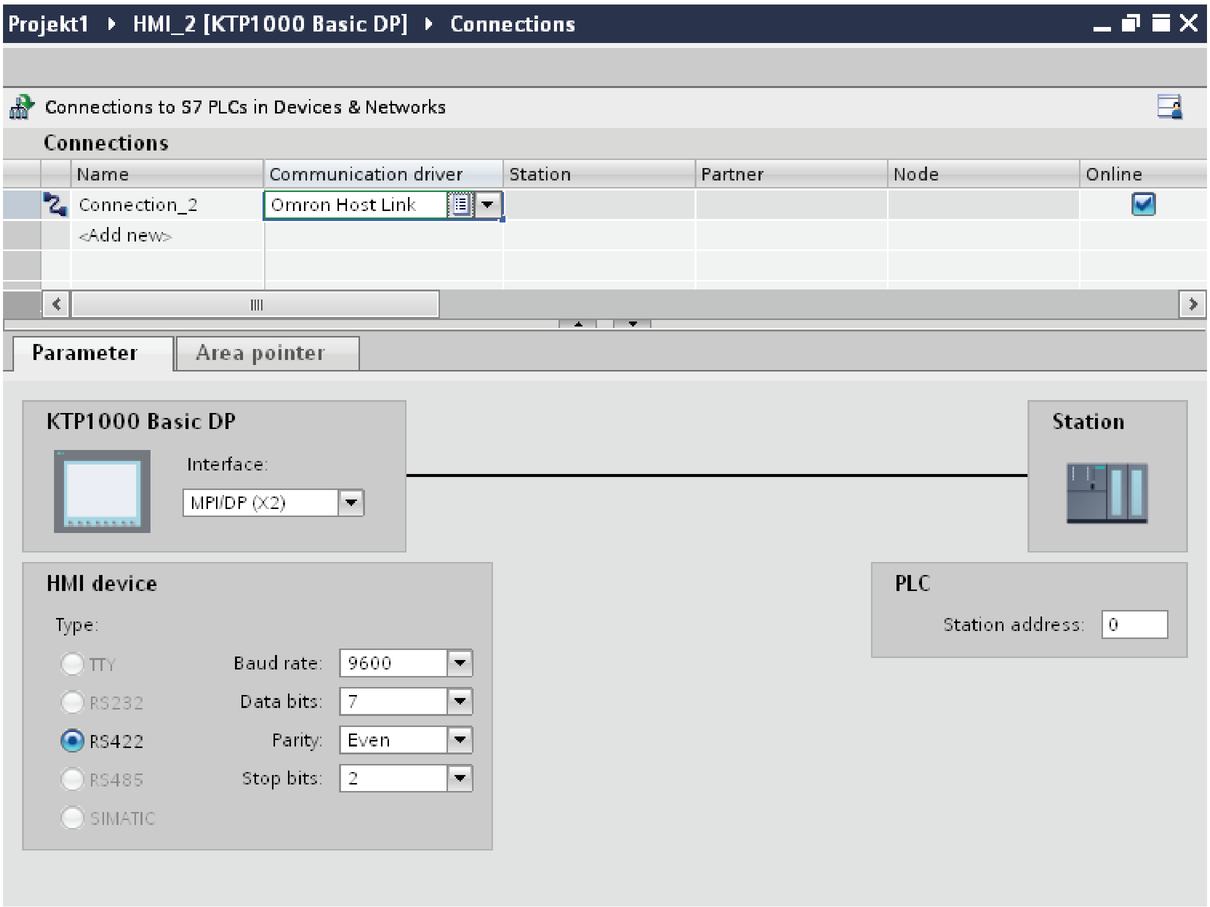
- A project is open.
- An HMI device has been created.

Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.
3. Double-click "<Add>" in the "Connections" editor.



4. Select the "Omron Host Link" driver in the "Communication driver" column.



Projekt1 ▸ HMI_2 [KTP1000 Basic DP] ▸ Connections

Connections to S7 PLCs in Devices & Networks

Connections

Name	Communication driver	Station	Partner	Node	Online
Connection_2	Omron Host Link				☑
<Add new>					

KTP1000 Basic DP

Interface: MPI/DP (X2)

HMI device

Type:

○ TTY
○ RS232
● RS422
○ RS485
○ SIMATIC

Baud rate: 9600

Data bits: 7

Parity: Even

Stop bits: 2

PLC

Station address: 0

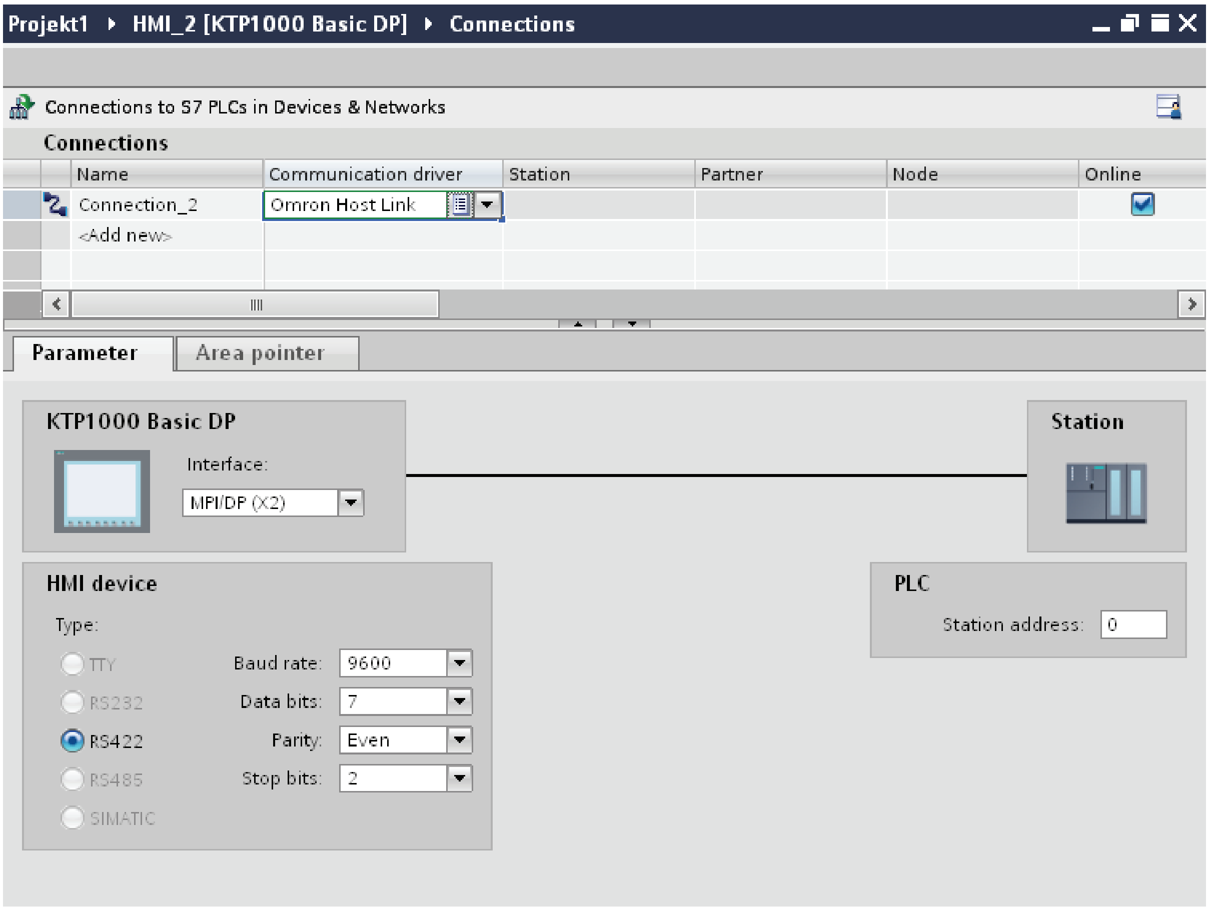
5. Select all necessary connection parameters for the interface in the inspector window under "Parameters".

Parameters for the connection (Omron Hostlink)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device" and "PLC" areas are available for assigning parameters according to the interface used.



Projekt1 ▸ HMI_2 [KTP1000 Basic DP] ▸ Connections

Connections to S7 PLCs in Devices & Networks

Connections

Name	Communication driver	Station	Partner	Node	Online
Connection_2	Omron Host Link				☑
<Add new>					

KTP1000 Basic DP

Interface: MPI/DP (X2)

Station

HMI device

Type:

○ TTY
○ RS232
● RS422
○ RS485
○ SIMATIC

Baud rate: 9600
Data bits: 7
Parity: Even
Stop bits: 2

PLC

Station address: 0

Parameters for the HMI device

You can select an interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

- Type

  Specifies the physical connection used.

- Baud rate

  For "Baud rate", you set the transmission speed of the HMI device to OMRON. A baud rate of 19200 or 9600 can be selected for the communication.

- Data bits

  For "Data bits", you can choose between "7 bits" and "8 bits".

- Parity

  For "Parity", you can choose from "None", "Even", and "Odd".

- Stop bits

  For "Stop bits", you can choose between 1 and 2 bits.

Parameters for the PLC

- Station address

  For "Station address", set the station number of the connected PLC.

Connecting HMI device to PLC

Connections via Omron Host Link

Connection

The connection of the HMI device to an OMRON PLC is limited primarily to the physical connection of the HMI device. Special blocks for the connection are not required in the PLC.

Connection cable

The following connecting cables are available to connect the HMI device to an Omron PLC.

Interface to the HMI device	**Omron PLC**			
	RS232, 9-pin	**RS232 I/O port**	**RS422, 9-pin**	**RS422, terminals/pins**
RS232, 9-pin	PP1	Programming cable (standard cable of Omron)	—	—
RS232 via converter	—	—	—	Multi-point cable 1
RS422, 9-pin	—	—	PP2	Multi-point cable 2

Refer to the relevant device manual to determine which HMI device interface is to be used.

Communication types

Approved communication types

The connection from a HMI device to an OMRON-CPU with the Omron Host Link protocol via RS232 and via RS 422 is system-tested and approved by Siemens AG.

This concerns the following CPU types:

- CP1x (CP1L, CP1H, CP1E)
- CJ1x(CJ1M, CJ1H, CJ1G)
- CJ2H
- CS1x(CS1G, CS1H, CS1D)
- CPM2C

Note

Only the following CPU types have been tested and released for Basic Panels, TP 177A and OP 77A:

- CP1x (CP1L, CP1H, CP1E)
- CJ1x (CJ1M, CJ1H, CJ1G)

Multipoint connection

A multipoint connection to the up to 4 approved OMRON PLCs in a RS422-four-wire connection can be implemented with communication modules on the PLCs and is system-tested and approved by Siemens AG.

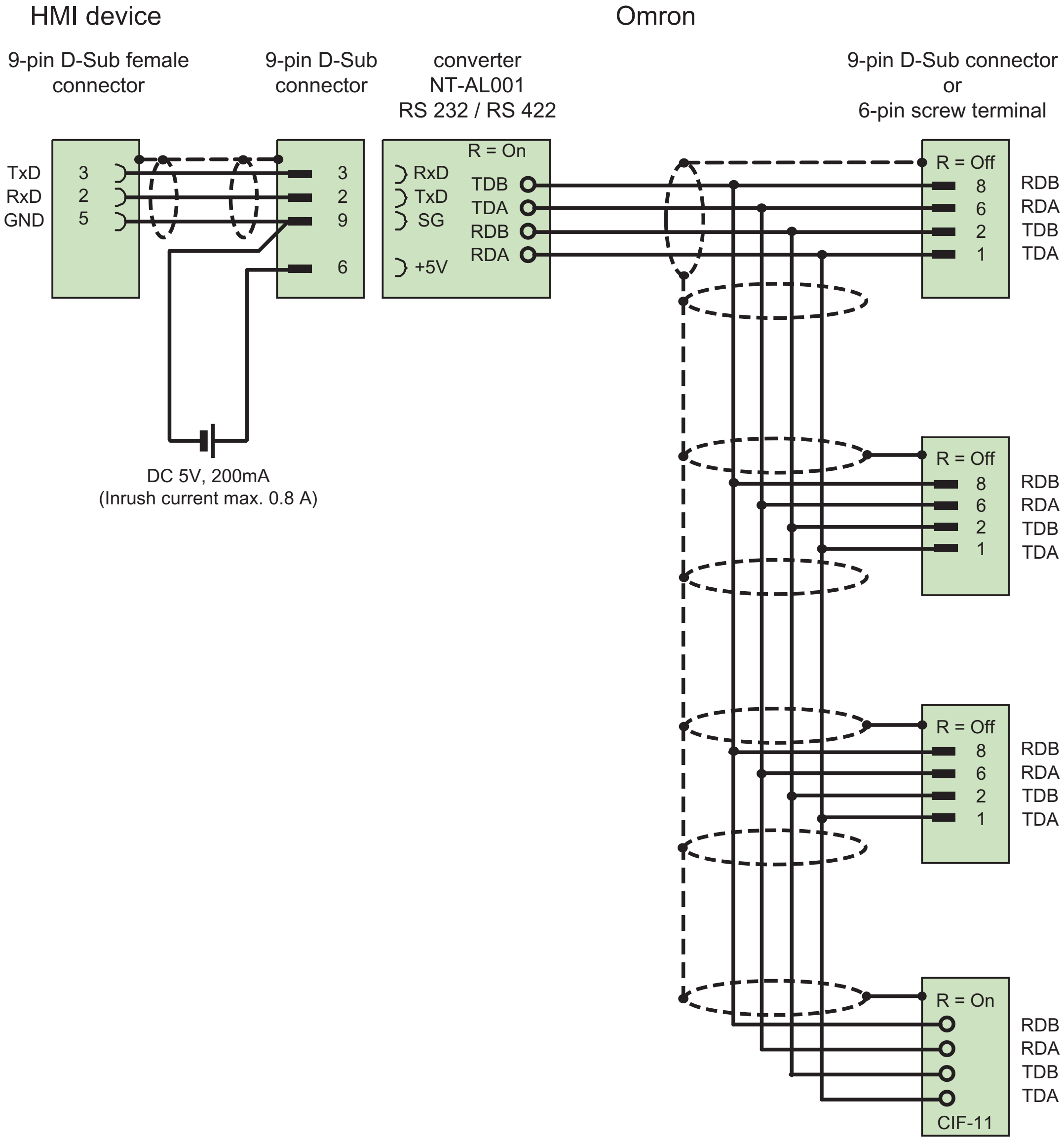
Note

The HMI device can only be operated as a master. Exactly one master is possible in the RS422-four-wire-Multidrop connection.

Connecting cable

Connecting cable MP1, RS-232, over converter, for Omron

Multipoint cable 1: MP/TP/PC > PLC

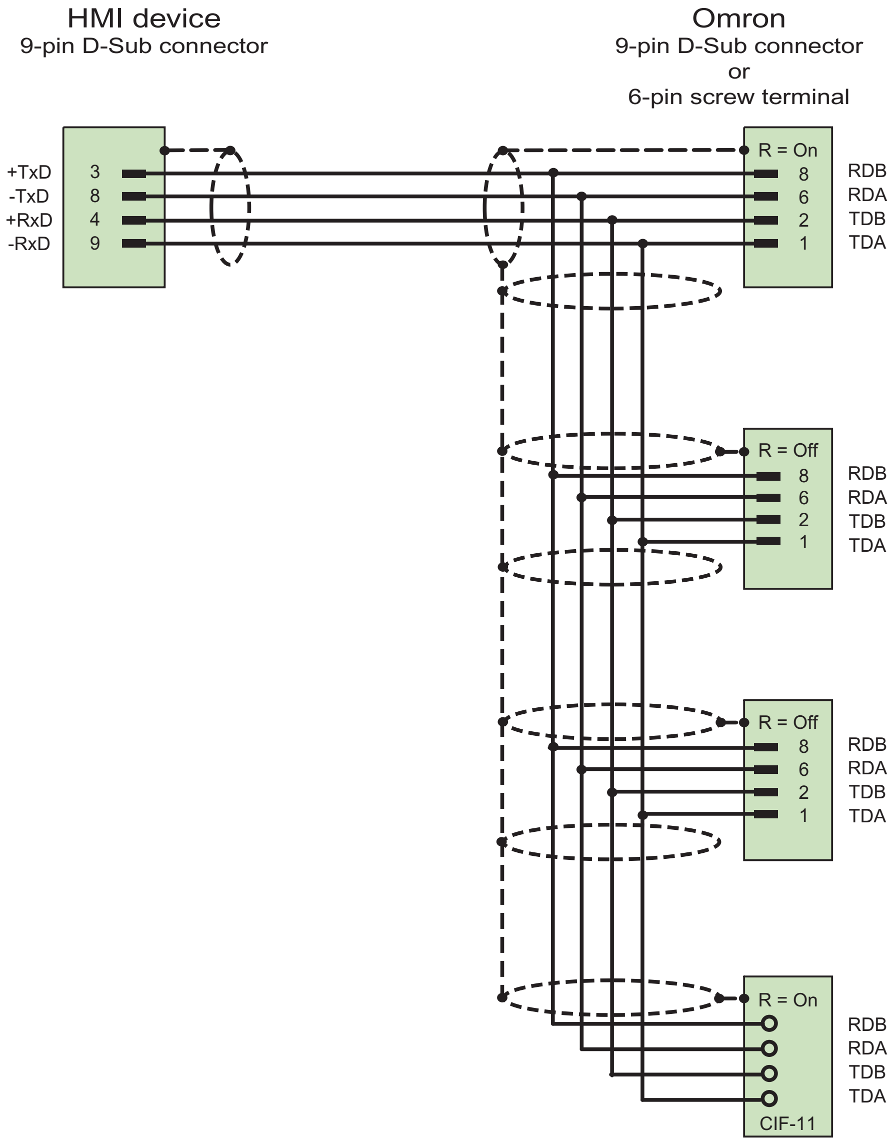


[1] Inrush current max. 0.8 A

shielded, max. length 500 m

Connecting cable MP2, RS-422, for Omron

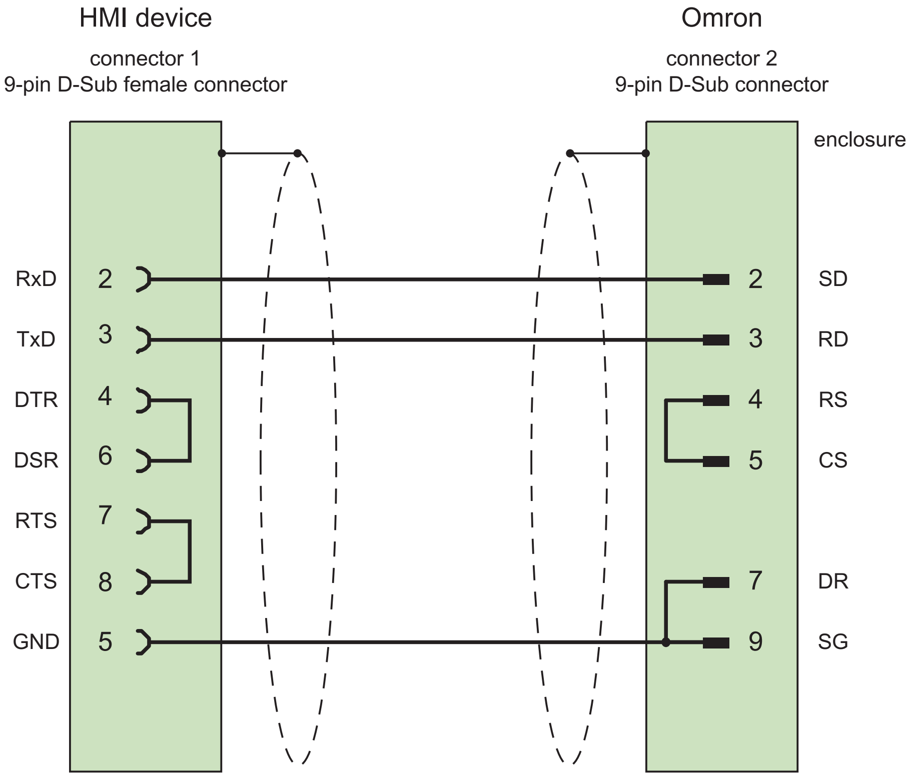
Multipoint cable 2: RS422, MP/TP/PC > SPS_



shielded, max. length 500 m

Connecting cable PP1, RS-232, for Omron

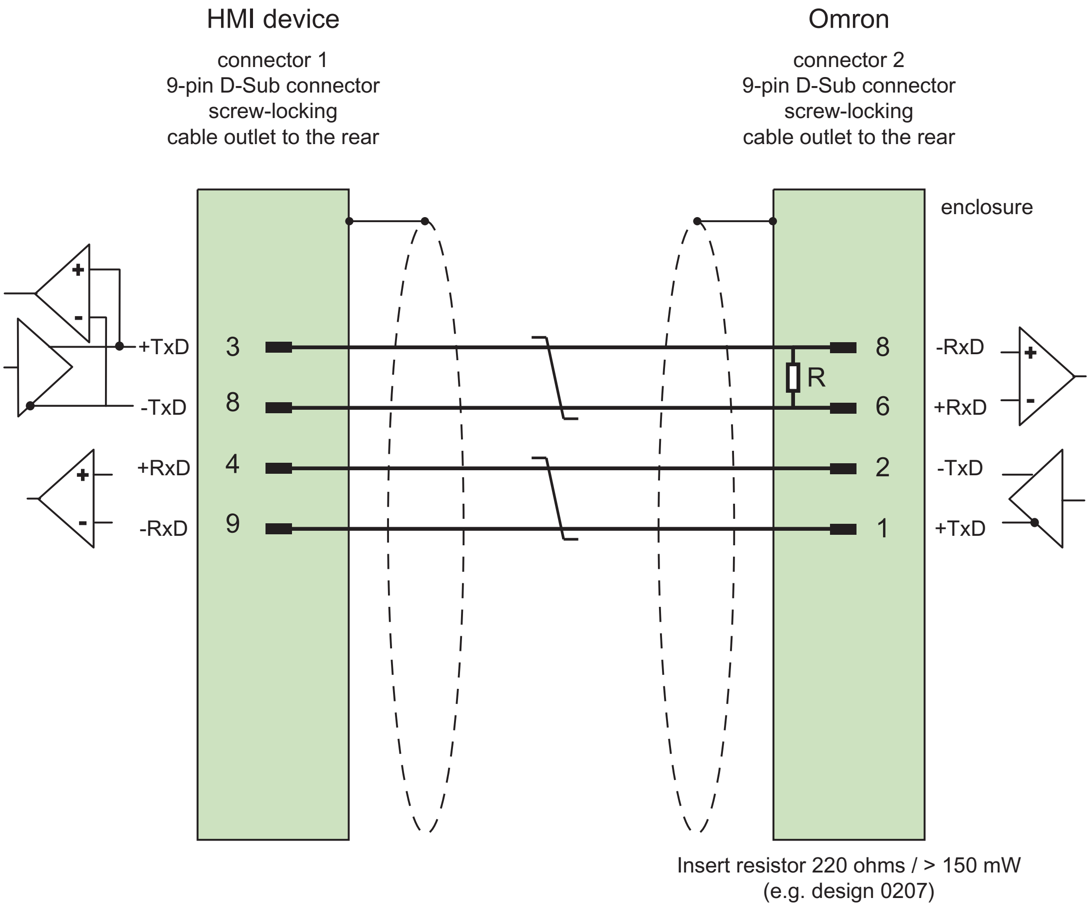
Point-to-point cable PP1, PC/TP/OP - PLC



shielded, max. length 15 m

Connecting cable PP2, RS-422, for Omron

Point-to-point cable PP2, RS-422



shielded, max. length 500 m

Performance features of communication

Permissible data types for Omron Host Link

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Operand type	Length
Bool	I/O, HR, AR, LR, DM, T/C bit, CPU status	1-bit
Byte	CPU type	1 byte
DInt	HR, AR, LR, DM	4 bytes
Int	I/O, HR, AR, LR, DM, T/C Val	2 bytes
Real	HR, DM	4 bytes
String	HR, AR, LR, DM	0 to 80 characters
UDInt	HR, AR, LR, DM	4 bytes
UInt	I/O, HR, AR, LR, DM, T/C Val	2 bytes

Note

Read and write operations of all data areas in the OMRON PLC can only be reliably carried out in "STOP" or "MONITOR" mode.

"I/O" refers either to the IR/SR area or the CIO area depending on the PLC series. The operand types "LR", "HR" and "AR" are not available in all PLC series.

Note

Note the following for write accesses:

For the "Bool" data type with the operand types "I/O", "HR", "AR", "LR" and "DM", the entire word is written back into the PLC when the specified bit is changed. There is no check to determine whether any other bits in the word have changed. As a result, the PLC only has read access to the specified word.

Operand type old PLC	Operand type CS and CJ PLC
CPU Status	CPU Status
I/O	CIO
HR	H Range 0-511
AR	A

Operand type old PLC	Operand type CS and CJ PLC
LR	n/a 1)
DM	D
T/C	T/C
CPU type	CPU type

1) You do not get an error message when you read or write the LR area in the following PLCs

- CS
- CJ
- CP

Supported CPU types for Omron Host Link

CPU types

The following CPU types are supported in the configuration of the Omron Host Link communication driver.

- CP1
  - CP1L
  - CP1H
  - CP1E
- CJ1
  - CJ1M
  - CJ1H
  - CJ1G
- CJ2
  - CJ2H
- CS1
  - CS1G
  - CS1H
  - CS1D
- CPM
  - CPM2C

Addressing in Omron Host Link

Addressing of PLCs in Omron Host Link

In PLCs of the series CS, CP and CJ, the timers 0-4095 are addressed with T/C 0-2047.

The counters 0-4095 must be addressed with an offset of 2048 (T/C 2048-4095 correspond to the counters 0-2047). Counters and timers with addresses > 2047 cannot be addressed via Host Link.

Counters and timers with addresses > 2047 cannot be addressed via Host Link.

Example:

If you want to address counter C20, you must address T/C 20+2048 = T/C 2068.

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.

   The project is compiled automatically.

   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Data exchange

Area pointers for Omron

Area pointers in connections via Omron communication drivers

You use an area pointer to access a data area in the PLC.

For more detailed information on area pointers and their configuration, refer to Section:

"Data exchange using area pointers".

Special features of connections via Omron Host Link

Area pointers can only be created in the following "File types": "DM", "I/O", "HR", "AR", and "LR".

See also

Data exchange using area pointers (Page 2676)

Trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out time-triggered for Basic Panels.

For additional information see:

Configuring trend displays for values from the PLC (Page 2155)

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration.

Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Alarms

Configuring alarms

Configuring alarms for non-integrated connections

Several steps are necessary to configure alarms such as warnings, error messages and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with alarms (Page 2167)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with an Omron communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
CP1, CJ1, CJ2, CS1, CPM	Uint, int	UInt, Int, UDInt, DInt

How the bit positions are counted

For connections with an Omron communication driver, the following counting method applies:

How the bit positions are counted	Left byte								Right byte							
In Omron PLCs	15							8	7							0
In WinCC you configure:	15							8	7							0

Only tags for the "DM", "I/O", "HR", "AR", and "LR" file types are allowed for use as a trigger tag for discrete alarms.

Configuring discrete alarms

Use arrays for discrete alarms and append each individual alarm to one bit of the array tags themselves and not to the individual subelements.

Only tags for the "DM", "I/O", "HR", "AR", LR" areas and the "Int" and "UInt" file types are permitted for discrete alarms and arrays.

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

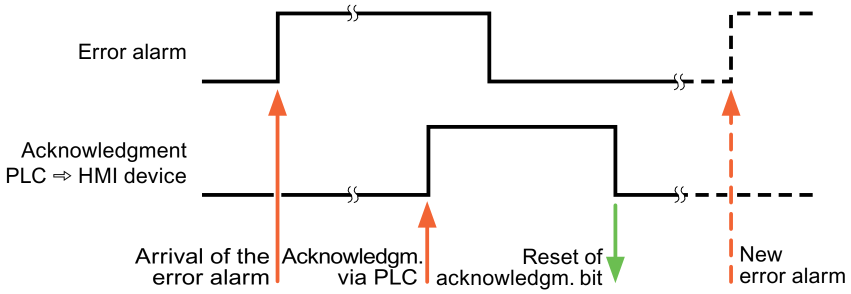
Acknowledgment by the PLC

In "Write acknowledgment tag", you configure the tag or the array tag and the bit number based on which the HMI device can recognize an acknowledgment by the PLC.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In "Read acknowledgment tag", you configure the tag or the array tag and the bit number that is written to the PLC after acknowledgment from the HMI device. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

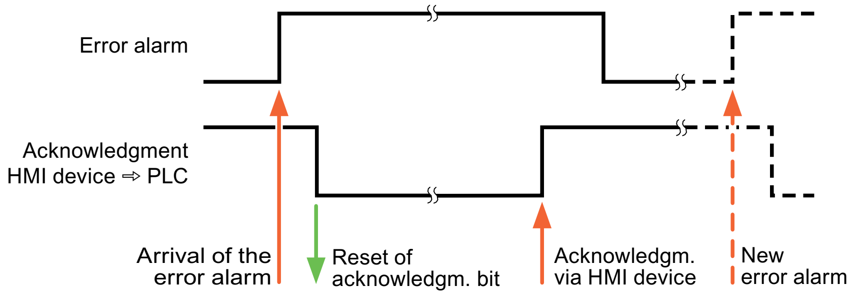
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.


Error alarm
Acknowledgment
HMI device ⇨ PLC
Arrival of the error alarm
Reset of acknowledgm. bit
Acknowledgm. via HMI device
New error alarm

10.8.8.4 Data exchange using area pointers

General information on area pointers

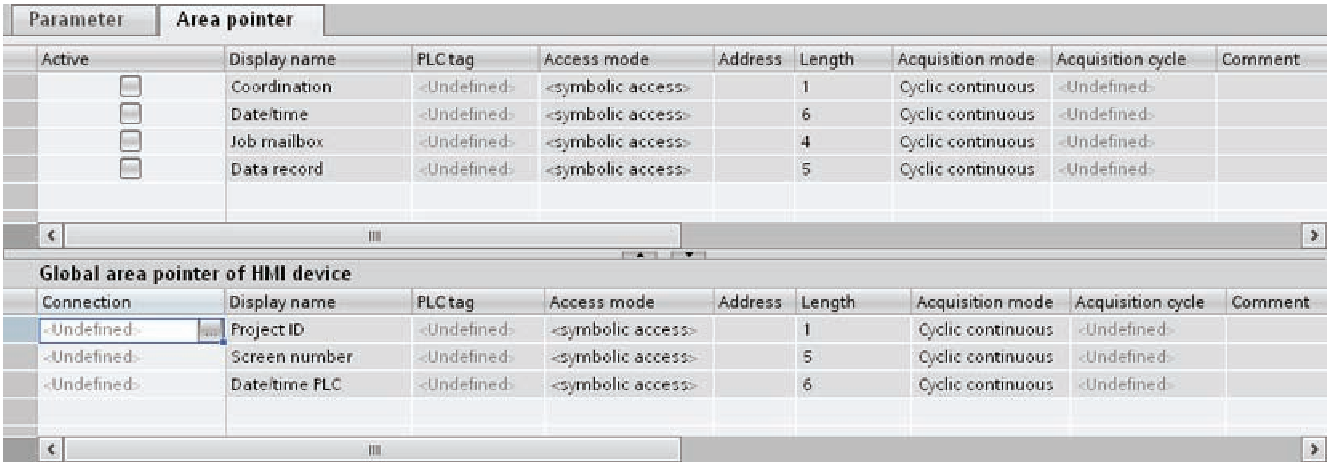
Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

Configuration of area pointers

Before you use the area pointer, you enable it in "Connections ▸ Area pointers". You then assign the area pointer parameters.



Active	Display name	PLC tag	Access mode	Address	Length	Acquisition mode	Acquisition cycle	Comment
□	Coordination	<Undefined>	<symbolic access>		1	Cyclic continuous	<Undefined>	
□	Date/time	<Undefined>	<symbolic access>		6	Cyclic continuous	<Undefined>	
□	Job mailbox	<Undefined>	<symbolic access>		4	Cyclic continuous	<Undefined>	
□	Data record	<Undefined>	<symbolic access>		5	Cyclic continuous	<Undefined>	

Global area pointer of HMI device

Connection	Display name	PLC tag	Access mode	Address	Length	Acquisition mode	Acquisition cycle	Comment
<Undefined>	Project ID	<Undefined>	<symbolic access>		1	Cyclic continuous	<Undefined>	
<Undefined>	Screen number	<Undefined>	<symbolic access>		5	Cyclic continuous	<Undefined>	
<Undefined>	Date/time PLC	<Undefined>	<symbolic access>		6	Cyclic continuous	<Undefined>	

- Active

  Enables the area pointer.
- Pointer name

  Name of the area pointer specified by WinCC.
- PLC tag

  Here you select the PLC tag or the tag array that you have configured as the data area for the area pointer.
- Address

  No address is entered into this field because of the symbolic access.
- Length

  WinCC specifies the length of the area pointer.

- Acquisition cycle

  You specify the acquisition cycle in this field for area pointers that are read by the HMI device. Note that a very short acquisition time may have a negative impact on HMI device performance.

- Comment

  Enter a comment, for example, to describe the purpose of the area pointer.

Accessing data areas

Accessing data areas

The following table shows how HMI devices and PLCs access individual data areas for read (R) or write (W) operations.

Data area	Required for	HMI device	PLC
Screen number	Evaluation by the PLC in order to determine the active screen.	W	R
Data record	Transfer of data records with synchronization	R/W	R/W
Date/time	Transfer of the date and time from the HMI device to the PLC	W	R
Date/time PLC	Transfer of the date and time from the PLC to the HMI device	R	W
Coordination	Requesting the HMI device status in the PLC program	W	R
Project ID	Runtime checks for consistency between the WinCC project ID and the project in the PLC	R	W
Job mailbox	Triggering of HMI device functions by the PLC program	R/W	R/W

"Screen number" area pointer

Function

The HMI device saves information about the screen called on the HMI device to the "Screen number" area pointer.

This allows the transfer of the current screen contents from the HMI device to the PLC. The PLC can trigger specific reactions such as the call of a different screen.

Use

Configure and enable the area pointer in "Communication > Connections" before you put it into use. You can create only **one** instance of the "Screen number" area pointer and only on **one** PLC.

The screen number is always transferred to the PLC when a new screen is activated or when the focus within a screen changes from one screen object to another.

Structure

The area pointer is a data area in the memory of the PLC with a fixed length of 5 words.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1. word	Current screen type															
2. word	Current screen number															
3. word	Reserved															
4th word	Current field number															
5. word	Reserved															

- Current screen type

  "1" for root screen or
  "4" for permanent window
- Current screen number

  1 to 32767
- Current field number

  1 to 32767

"Date/time" area pointer

Function

This area pointer is used to transfer the date and time from the HMI device to the PLC.

The PLC writes control job "41" to the job mailbox.

When it evaluates the control job, the HMI device writes its current date and the time in the data area configured in the "Date/time" area pointer. All definitions are coded in BCD format.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

The date/time data area has the following structure:

Data word	Left byte								Right byte								
	15							8	7							0	
n+0	Reserved								Hour (0 to 23)								Time
n+1	Minute (0 to 59)								Second (0 to 59)								
n+2	Reserved								Reserved								
n+3	Reserved								Weekday (1 to 7, 1=Sunday)								Date
n+4	Day (1 to 31)								Month (1 to 12)								
n+5	Year (80 to 99/0 to 29)								Reserved								

Note

Note that when you enter the year, values 80 to 99 result in years 1980 through 1999 and the values 0 to 29result in the years 2000 through 2029.

"Date/time PLC" area pointer

Function

This area pointer is used to transfer the date and time from the PLC to the HMI device. Use this area pointer if the PLC is the time master.

The PLC loads the data area of the area pointer. All definitions are coded in BCD format.

The HMI device reads the data cyclically within the configured acquisition cycle and synchronizes itself.

Note

Set an acquisition cycle of sufficient length for the date/time area pointer in order to avoid any negative impact on HMI device performance.
Recommended: Acquisition cycle of 1 minute if your process can handle it.

The "Date/time PLC" data area has the following structure:

<table>
<tr><th rowspan="2">Data word</th><th colspan="3">Left byte</th><th colspan="3">Right byte</th></tr>
<tr><th>15</th><th>......</th><th>8</th><th>7</th><th>......</th><th>0</th></tr>
<tr><td>n+0</td><td colspan="3">Year (80 to 99/0 to 29)</td><td colspan="3">Month (1 to 12)</td></tr>
<tr><td>n+1</td><td colspan="3">Day (1 to 31)</td><td colspan="3">Hour (0 to 23)</td></tr>
<tr><td>n+2</td><td colspan="3">Minute (0 to 59)</td><td colspan="3">Second (0 to 59)</td></tr>
<tr><td>n+3</td><td colspan="3">Reserved</td><td colspan="2">Reserved</td><td>Weekday
(1 to 7,
1=Sunday)</td></tr>
<tr><td>n+4 [1]</td><td colspan="3">Reserved</td><td colspan="3">Reserved</td></tr>
<tr><td>n+5 [1]</td><td colspan="3">Reserved</td><td colspan="3">Reserved</td></tr>
</table>

[1] The two data words must exist in the data area to ensure that the data format matches WinCC and to avoid reading false information.

Note

Note that when you enter the year, values 80 to 99 result in years 1980 through 1999 and the values 0 to 29result in the years 2000 through 2029.

"Coordination" area pointer

Function

The "Coordination" area pointer is used to implement the following functionality:

- Detecting the startup of the HMI device in the control program
- detection in the control program of the current HMI device operating mode
- detection in the control program of the HMI devices ready to communicate state

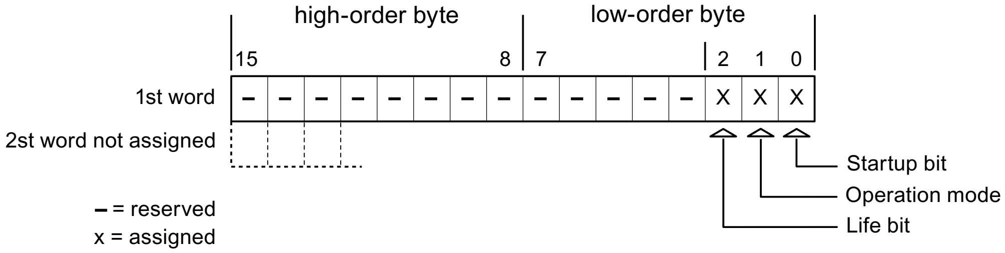
The "Coordination" area pointer has a length of one word.

Use

Note

The HMI device always writes the entire coordination area when updating the area pointer. The control program may not make changes to the coordination area for this reason.

Assignment of bits in the "Coordination" area pointer



Startup bit

The startup bit is set briefly to "0" by the HMI device during startup. It sets the bit permanently to "1" when startup is completed.

Operating mode

The operating mode bit is set to 1 as soon as the user switches the HMI device offline. The state of the operating mode bit is "0" during normal operation of the HMI device. You can determine the current operating mode of the HMI device by reading this bit.

Life bit

The HMI device inverts the life bit at intervals of approximately one second. You can check whether or not the connection to the HMI device is still up by querying this bit in the control program.

"Project ID" area pointer

Function

You can check whether the HMI device is connected to the correct PLC at the start of runtime. This check is important when operating with several HMI devices.

The HMI device compares a value stored on the PLC with the value specified in configuration. This ensures compatibility of configuration data with the control program. If discrepancy is detected, a system event is displayed on the HMI device and runtime is stopped.

Use

To use this area pointer, set up the following during the configuration:

- Define the version of configuration. Possible values between 1 and 255.

  You enter the version in the editor "Runtime settings > General" in the "Identification" area.
- Data address of the value for the version that is stored in the PLC:

  You enter the data address in the editor "Communication > Connections".

Connection failure

A connection failure to a device on which the "project ID" area pointer is configured results in all the other connections in the project being switched to "offline".

This behavior has the following prerequisites:

- You have configured several connections in a project.
- You are using the "project ID" area pointer in at least one connection.

Causes which may set connections "offline":

- The PLC is not available.
- The connection has been switched offline in the engineering system.

"PLC job" area pointer

Function

The PLC can use the job mailbox to transfer jobs to the HMI device to trigger corresponding actions on the HMI device. These functions include, for example:

- Display screen
- Set date and time

Data structure

The first word of the job mailbox contains the job number. Depending on the job mailbox, up to three parameters can be transferred.

Word	Left byte	Right byte
n+0	0	Job number
n+1	Parameter 1	
n+2	Parameter 2	
n+3	Parameter 3	

The HMI device evaluates the job mailbox if the first word of this job is not equal to zero. This means that the parameters must be entered in the job mailbox first, followed by the job number.

When the HMI device accepts the job mailbox, the first word is set to 0 again. The execution of the job mailbox is generally not completed at this point in time.

Job mailboxes

All job mailboxes and their parameters are listed below. The "No." column contains the job number of the job mailbox. Job mailboxes can only be triggered by the PLC when the HMI device is online.

Note

Please note that not all HMI devices support job mailboxes.

No.	Function	
14	**Setting the time (BCD coded)**	
	Parameter 1	Left byte: - Right byte: hours (0-23)
	Parameter 2	Left byte: minutes (0-59) Right byte: seconds (0-59)
	Parameter 3	-

No .	Function	
14	Setting the time (BCD coded)	
15	Setting the date (BCD coded)	
	Parameter 1	Left byte: - Right byte: weekday (1-7: Sunday-Saturday)
	Parameter 2	Left byte: day (1-31) Right byte: month (1-12)
	Parameter 3	Left byte: year
23	User logon	
	Logs the user on with the name "PLC user" at the HMI device with the group number transferred in Parameter 1. The logon is possible only when the transferred group number exists in the project.	
	Parameter 1	Group number 1 to 255
	Parameter 2, 3	-
24	User logoff	
	Logs off the current user. (The function corresponds to the "logoff" system function)	
	Parameter 1, 2, 3	-
40	Transfer date/time to PLC	
	(in the S7 format DATE_AND_TIME) An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
41	Transfer date/time to PLC	
	(In OP/MP format) An interval of at least 5 seconds must be maintained between successive jobs in order to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
46	Update tags	
	Causes the HMI device to read the current value of the tags•from the PLC whose update ID matches the value transferred in parameter 1. (Function corresponds to the "UpdateTag" system function.)	
	Parameter 1	1 - 100
49	Delete alarm buffer	
	Deletes all analog alarms and discrete alarms of the "Warnings" class from the alarm buffer.	
	Parameter 1, 2, 3	-
50	Delete alarm buffer	
	Deletes all analog alarms and discrete alarms of the "Errors" class from the alarm buffer.	
	Parameter 1, 2, 3	-
51	Screen selection [1)]	
	Parameter 1	Screen number
	Parameter 2	-
	Parameter 3	Field number
69	Read data record from PLC	

No.	Function	
14	Setting the time (BCD coded)	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	0: Do not overwrite existing data record 1: Overwrite existing data record
70	Write data record to PLC	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	-

[1] OP 73, OP 77A and TP 177A HMI devices also execute the "Screen selection" job mailbox if the on-screen keyboard is active.

"Data record" area pointer

"Data mailbox" area pointer

Function

When data records are transferred between the HMI device and PLC, both partners access common communications areas on the PLC.

Data transfer types

There are two ways of transferring data records between the HMI device and PLC:

- Transfer without synchronization
- Transfer with synchronization over the data record

Data records are always transferred directly. That is, the tag values are read from an address or written to an address configured for this tag directly, without redirecting the values by means of interim memory.

Initiating the transfer of data records

There are three ways of triggering the transfer:

- Operator input in the recipe view
- Job mailboxes

  The transfer of data records can also be triggered by the PLC.
- Triggering by configured functions

If the transfer of data records is triggered by a configured function or by a job mailbox, the recipe view on the HMI device remains operable. The data records are transferred in the background.

Simultaneous processing of several transfer requests is, however, not possible. In this case, the HMI device rejects the other transfer requests with a system event.

Transfer without synchronization

If you select asynchronous transfer of data records between the HMI device and PLC, there is no coordination over the common data areas. It is therefore unnecessary to set up a data area during configuration.

Asynchronous data record transfer can be a useful alternative, for example, when:

- The system is capable of excluding the risk of uncontrolled overwriting of data by the communication peer.
- The PLC does not require information about the recipe number and data record number.
- The transfer of data records is triggered by the operator of the HMI device.

Reading values

When a read job is triggered, the values are read from the PLC addresses and transferred to the HMI device.

- Triggering by the operator in the recipe view:

  The values are downloaded to the HMI device. You can then process, edit, or save these values, for example.
- Triggering by a function or job mailbox:

  The values are saved immediately to the data volume.

Writing values

When a write job is triggered, the values are written to the PLC addresses.

- Triggering by the operator in the recipe view:

  The current values are written to the PLC.
- Triggering by a function or job mailbox:

  The current values are written to the PLC from the data medium.

Transfer with synchronization

If you select synchronous transfer, both communication partners set status bits in the common data area. You can use this mechanism to prevent uncontrolled overwriting of data in either direction in your control program.

Application

Synchronous data record transfer can be a useful solution, for example, when:

- The PLC is the "active partner" in the transfer of data records.
- The PLC evaluates the information about the recipe number and data record number.
- The transfer of data records is triggered by means of a Job mailbox.

Requirements

In order to synchronize transfer of data records between the HMI device and the PLC, the following requirements must be met during configuration:

- An area pointer has been set up: "Communication > Connections" editor in "Area pointer".
- The PLC with which the HMI device synchronizes transfer of data records is specified in the recipe:

  "Recipes" editor in the inspector window the option "Coordinated transfer of data records" under "General > Synchronization > Settings"

Structure of the data area

The data area has a fixed length of 5 words. Structure of the data area:

	15		0
1. Word	Current recipe number (1 - 999)		
2. Word	Current data record number (0 - 65535)		
3. Word	Reserved		
4. Word	Status (0, 2, 4, 12)		
5. Word	Reserved		

- Status

  The status word (word 4) can adopt the following values:

Value		Meaning
Decimal	Binary	
0	0000 0000	Transfer permitted, data record free
2	0000 0010	Transferring.
4	0000 0100	Transfer completed without error
12	0000 1100	Transfer completed with error

Sequence of a transfer started by the operator in the recipe display

Reading from the PLC started by the operator in the recipe view

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe number to be read and the status "Transferring" in the data record and sets the data record number to 0.	Abort with system event.
3	The HMI device reads the values from the PLC and displays them in the recipe view. If the recipes have synchronized tags, the values from the PLC are also written to the tags.	
4	The HMI device sets the status "Transfer completed."	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC started by the operator in the recipe view

Step	Action	
	Check: Status word = 0?	
1	Yes	No
	The HMI device enters the recipe and data record number to be written and the status "Transferring" in the data record.	Abort with system event.
2	The HMI device writes the current values to the PLC. If the recipes have synchronized tags, the changed values are synchronized between the recipe view and tags and then written to the PLC.	
3	The HMI device sets the status "Transfer completed."	
4	If required, the control program can now evaluate the transferred data.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Sequence of the transfer triggered by a PLC job

The transfer of data records between the HMI device and the PLC can be initiated by either one of these stations.

The two job mailboxes No. 69 and No. 70 are available for this type of transfer.

No. 69: Read data record from PLC ("PLC → DAT")

Job mailbox no. 69 transfers data records from the PLC to the HMI device. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	69
Word 2	Recipe number (1-999)	
Word 3	Data record number (1 to 65535)	
Word 4	Do not overwrite existing data record: 0 Overwrite existing data record: 1	

No. 70: Write data record to PLC ("DAT → PLC")

Job mailbox No. 70 transfers data records from the HMI device to the PLC. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	70
Word 2	Recipe number (1-999)	
Word 3	Data record number (1 to 65535)	
Word 4	—	

Sequence when reading from the PLC with job mailbox "PLC → DAT" (no. 69)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data record.	Abort without return message.

Step	Action	
3	The HMI device reads the values from the PLC and saves these to the data record defined in the job mailbox.	
4	• If "Overwrite" was selected in the job, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "Do not overwrite" was selected in the job, and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data record.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Sequence of writing to the PLC with job mailbox "DAT → PLC" (no. 70)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data record.	Abort without return message.
3	The HMI device fetches the values of the data record specified in the function from the data medium and writes the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Sequence of the transfer when triggered by a configured function

Reading from the PLC using a configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data record.	Abort with system event.
3	The HMI device reads the values from the PLC and stores them in the data record specified in the function.	

Step	Action	
4	• If "Yes" was selected for the "Overwrite" function, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "No" was selected for the "Overwrite" function and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data record.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC by means of configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data record.	Abort with system event.
3	The HMI device fetches the values of the data record specified in the function from the data medium and transfers the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Possible causes of error when transferring data records

Possible causes of error

The section below shows possible error causes which lead to the cancellation of data record transfer:

- Tag address not set up on the PLC
- Overwriting data records not possible
- Recipe number does not exist
- Data record number does not exist

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Reaction to an aborted transfer due to errors

If the transfer of data records is aborted due to errors, the HMI device reacts as follows:

- Triggering by the operator in the recipe view

  Information in the status bar of the recipe view and output of system alarms
- Triggered by function

  Output of system alarms
- Triggering by job mailbox

  No return message on the HMI device

You can nonetheless evaluate the status of the transfer by querying the status word in the data record.

10.9 Using global functions

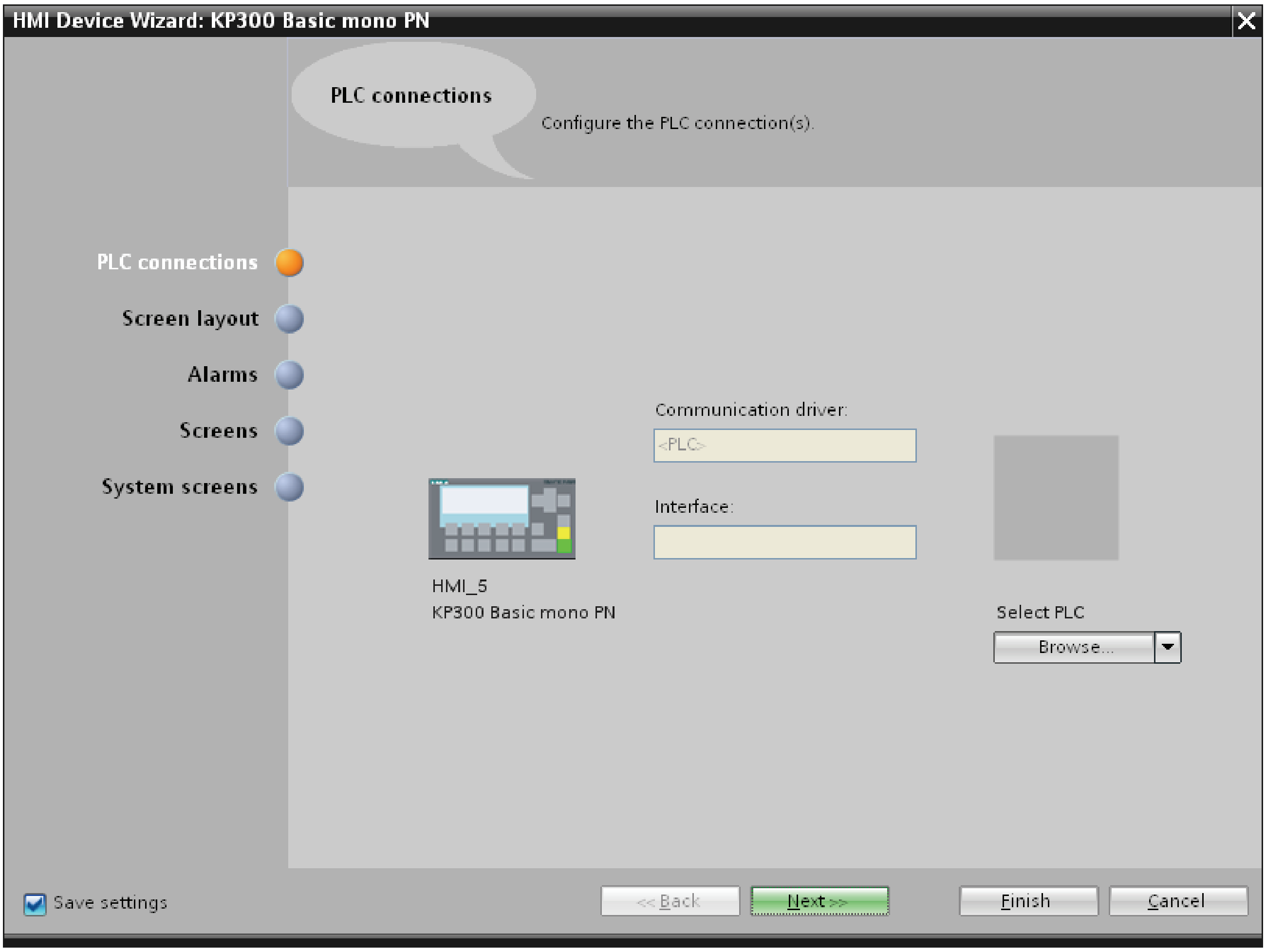
10.9.1 HMI device wizard basics

Introduction

The HMI device wizard will automatically start when you create a new HMI device in your project.

HMI device wizard

The HMI device wizard will guide you through each dialog step by step and help you set up a device. You use the HMI device wizard to specify the basic settings for your HMI device, such as screen layout and the connection to your PLC.


HMI Device Wizard: KP300 Basic mono PN

PLC connections

Configure the PLC connection(s).

Communication driver: <PLC>

Interface:

HMI_5
KP300 Basic mono PN

10.9.2 Importing and exporting project data

10.9.2.1 Importing and exporting project data

Introduction

WinCC enables you to export data from a project and import it into another project.

You export or import the following project data:

- Recipe data records
- Alarms
- Tags
- Text lists
- Project texts

Exporting and importing reduces the workload. Instead of creating new data records, you use data already created in previous projects.

Editing the export file

The following file formats are available for export and import depending on the editor:

- *.xlsx for alarms, tags, project texts and text lists
- *.csv for recipe data records

You can edit the import file in Excel, for example.

XLSX file format

XLSX format is a file format for Excel tables based on the Open XML format. XLSX files are optimized for Microsoft Excel 2007.

You can sort the columns as required in the XLSX file.

CSV file format

CSV stands for Comma Separated Value. In this format, the columns of the table that contain the names and the value of the entry are separated by semicolons. Each table row terminates with a line break. You can also open the CSV file for editing in Excel.

Importing project data

When project data is imported, the objects in the project are created.

The syntax of the import file is checked during import. The accuracy of the values imported and dependencies between the imported values are not checked.

Any errors found in the imported data are reported when the project is compiled.

See also

Exporting alarms (Page 2698)

Exporting tags (Page 2706)

Exporting text lists (Page 2711)

10.9.2.2 Importing and exporting recipes

Exporting recipes

Introduction

WinCC features an export function for exporting data records from recipes.

Requirements

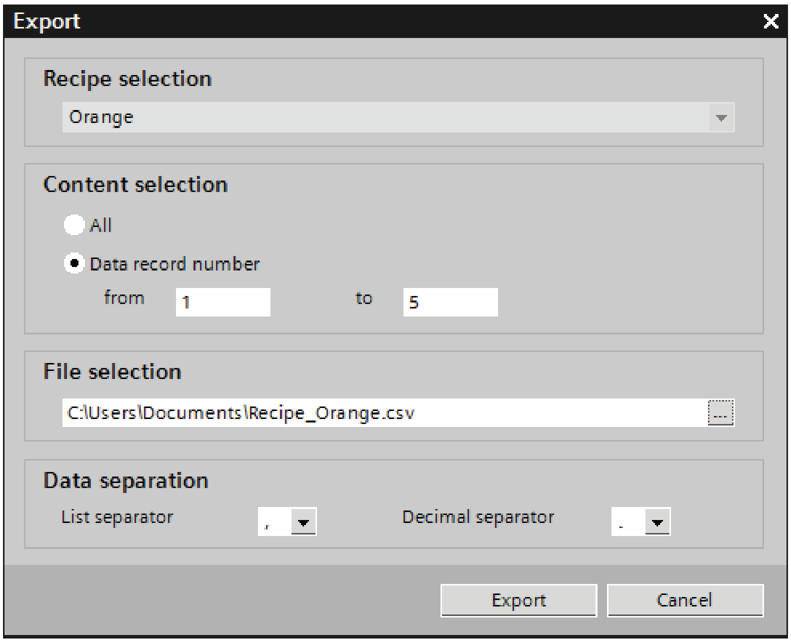
- The WinCC project for the export is open.
- Recipes have been created in a project.
- The "Recipes" editor is open.

Exporting recipes

1. In the "Recipes" editor, select the recipe with the data records you want to export.

2. Click ➡.

   The "Export" dialog box opens.



Recipe selection

Orange

Content selection

○ All

● Data record number

from 1 to 5

File selection

C:\Users\Documents\Recipe_Orange.csv

Data separation

List separator ,

Decimal separator .

   The selected recipe is shown under "Recipe selection".

3. Under "Content selection", specify if all or only selected data records are to be exported.
4. Under "File selection", specify the file in which the recipe data is to be stored.
5. Specify the list separator and decimal separator under "Data separation".
6. Click "Export."

   The export will start.

Result

The exported data has been written to a CSV file. The CSV file will be stored in the specified directory.

See also

Exporting alarms (Page 2698)

Importing and exporting project data (Page 2693)

Exporting tags (Page 2706)

Exporting text lists (Page 2711)

Importing recipes

Introduction

Recipes are identified by their name. The recipe name must therefore be unique. Open the import file in a simple text editor to check that it has the correct data structure.

Specify whether or not existing data records should be overwritten by records with the same name during the import.

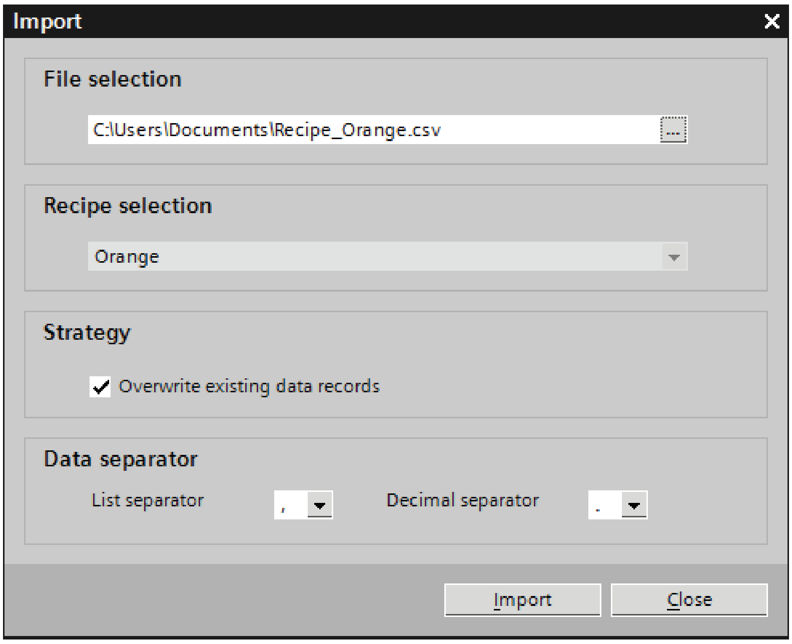
Requirements

- A CSV file containing at least one recipe has been created.
- The WinCC project for the import is open.
- The "Recipes" editor is open with at least one recipe.

Importing a recipe

1. In the "Recipes" editor, select the recipe with the data records you want to import.
2. Click .

   The "Import" dialog box opens.



Import

File selection

C:\Users\Documents\Recipe_Orange.csv

Recipe selection

Orange

Strategy

☑ Overwrite existing data records

Data separator

List separator ,

Decimal separator .

The selected recipe is shown under "Recipe selection".

3. Select the file you want to import under "File selection".
4. Under "Strategy", specify if existing data records should be overwritten by records of the same name.
5. Under "Data separation", select the list separator and the decimal separator to use in the CSV file.
6. Click "Import".

   The import will start.

Result

The data records are created in the selected recipe. Depending on the setting for "Strategy", existing data records are overwritten by records with the same name from the CSV file.

Existing data records with the same name will also be imported from the CSV file if you deactivate the "Overwrite existing data records" option.

Format of recipe data

Introduction

This section describes the required format of the file for the import of recipes. The file containing the data of the recipes must be available in "*.csv" format. :

Structure of recipe data

The structure of the import file is fixed. The following example shows the structure of a recipe containing two recipe elements, each with two data records:

```
List separator=<List separator>Decimal symbol=<Decimal
separator><List separator><Line break>
<Name of the recipe><List separator><List separator><Line break>
LANGID_<ID of the language><List separator>
<Display name, recipe element 1><List separator>
<Display name, recipe element 2><Line break>
<Number recipe><List separator>
<Recipe data record number 1><List separator>
<Recipe data record number 2><Line break>
<Tag recipe element 1><List separator>
<Recipe data record 1 value 1><List separator>
<Recipe data record 2 value 1><Line break>
<Tag recipe element 2><List separator>
<Recipe data record 1 value 2><List separator>
<Recipe data record 2 value 2><Line break>
```

ID of the language

Use the "Windows language ID" in decimal notation, e.g. "1033" for English. Additional information is available in the documentation for the Windows operating system.

See also

Exporting recipes (Page 2694)

10.9.2.3 Importing and exporting alarms

Exporting alarms

Introduction

WinCC has an export function for alarms.

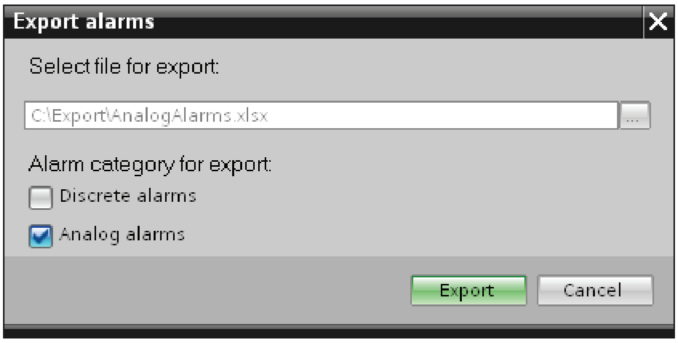
Requirements

- The WinCC project for the export is open.
- Alarms have been created in the project.
- The "HMI alarms" editor is open.

Exporting alarms

1. Click the button in "Discrete alarms" or "Analog alarms".

   The "Export" dialog box opens.

   Export alarms

   Select file for export:

   C:\Export\AnalogAlarms.xlsx [...]

   Alarm category for export:

   ☐ Discrete alarms

   ☑ Analog alarms

2. Click the "..." button and specify in which file the data are saved.
3. Specify whether you want to export "Disrete alarms" or "Analog alarms".
4. Click "Export". The export will start.

Result

The exported data has been written to an xlsx file. The xlsx file will be stored in the specified folder.

See also

Importing alarms (Page 2699)

Format of the analog alarm data (Page 2701)

Format of the discrete alarm data (Page 2704)

Exporting recipes (Page 2694)

Exporting tags (Page 2706)

Importing and exporting project data (Page 2693)

Importing alarms

Introduction

Alarms are identified by their alarm number. The alarm numbers must be unique in the analog and discrete alarm types. Alarms with redundant alarm numbers will be overwritten. An alarm without an existing alarm number is created.

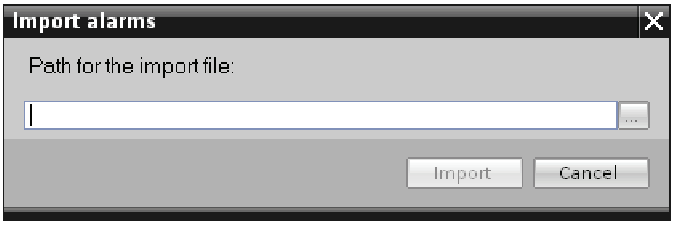
Any empty list entries for existing alarms contained in an xlsx file will be ignored for the purposes of the import. The entries of the existing alarms remain active and will not be replaced by empty ones.

Requirements

- An xlsx file with alarms has been created.
- The structure of the xlsx file meets the requirements.
- The WinCC project for the import is open.
- The "HMI alarms" editor is open.

Importing alarms

1. Click the button in "Discrete alarms" or "Analog alarms". The "Import" dialog box opens.



2. Click the "..." button and select the file that you want to import.
3. Click on the "Import" button. The import will start. A progress bar indicates the progress of the import operation.

Result

The corresponding alarms including alarm texts are created in WinCC on the basis of the import data. Alarms relating to the import operation are displayed in the output window. A log file is saved in the source directory of the import files. The log file has the same name as the respective import file but with the "*.xml" extension.

Check when importing the data whether there are any links to objects, for example, dynamic parameters such as tags.

- If an object with the same name already exists, the existing object is used.
- If no object of the same name yet exists, create an object with the relevant name or create a new link.

Note

The syntax of the import file is checked during xlsx file import. The meaning of the properties or dependencies between the properties is not checked. It is possible to assign a trigger tag of an incorrect type, such as string, to an alarm. An error will be reported during compilation.

See also

Exporting alarms (Page 2698)

Format of the analog alarm data

Introduction

This chapter describes the required format of the file for the import of analog alarms. The file containing the analog alarm data must be in "*.xlsx" format.

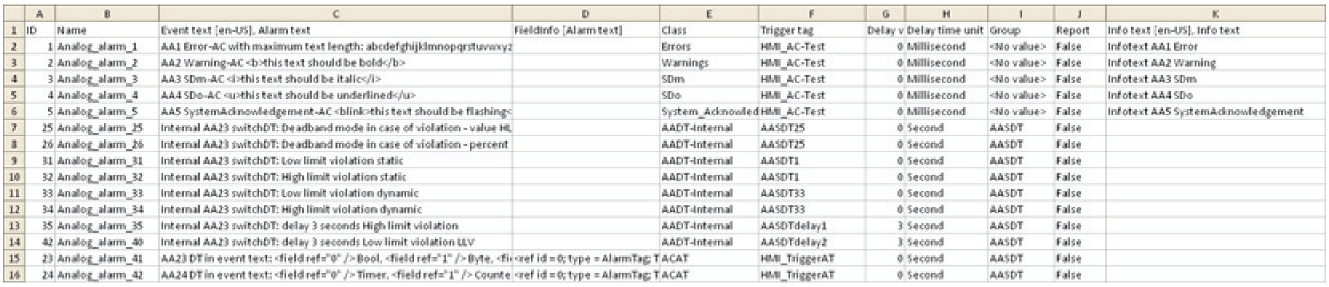
Structure of the alarm data

The import file in Microsoft Excel consists of a number of worksheets:

- Analog alarms(Analog alarms)
- Limits (Limits)

Each alarm is assigned a separate row in the import file. The import file with the analog alarms must be formatted as follows:

Example of the worksheet "Analog alarms"



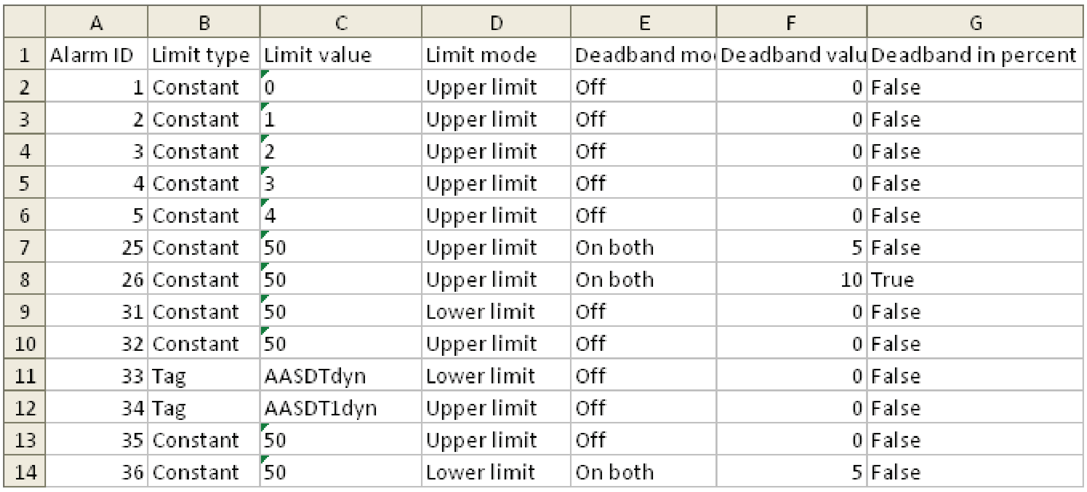
	A	B	C	D	E	F	G	H	I	J	K
1	ID	Name	Event text [en-US], Alarm text	FieldInfo [Alarm text]	Class	Trigger tag	Delay v	Delay time unit	Group	Report	Info text [en-US], Info text
2	1	Analog_alarm_1	AA1 Error-AC with maximum text length: abcdefghijklmnopqrstuvwxyz		Errors	HMI_AC-Test	0	Millisecond	<No value>	False	Infotext AA1 Error
3	2	Analog_alarm_2	AA2 Warning-AC <b>this text should be bold</b>		Warnings	HMI_AC-Test	0	Millisecond	<No value>	False	Infotext AA2 Warning
4	3	Analog_alarm_3	AA3 SDm-AC <i>this text should be italic</i>		SDm	HMI_AC-Test	0	Millisecond	<No value>	False	Infotext AA3 SDm
5	4	Analog_alarm_4	AA4 SDo-AC <u>this text should be underlined</u>		SDo	HMI_AC-Test	0	Millisecond	<No value>	False	Infotext AA4 SDo
6	5	Analog_alarm_5	AA5 SystemAcknowledgement-AC <blink>this text should be flashing<		System_Acknowled	HMI_AC-Test	0	Millisecond	<No value>	False	Infotext AA5 SystemAcknowledgement
7	25	Analog_alarm_25	Internal AA23 switchDT: Deadband mode in case of violation - value HL		AADT-Internal	AASDT25	0	Second	AASDT	False	
8	26	Analog_alarm_26	Internal AA23 switchDT: Deadband mode in case of violation - percent		AADT-Internal	AASDT25	0	Second	AASDT	False	
9	31	Analog_alarm_31	Internal AA23 switchDT: Low limit violation static		AADT-Internal	AASDT1	0	Second	AASDT	False	
10	32	Analog_alarm_32	Internal AA23 switchDT: High limit violation static		AADT-Internal	AASDT1	0	Second	AASDT	False	
11	33	Analog_alarm_33	Internal AA23 switchDT: Low limit violation dynamic		AADT-Internal	AASDT33	0	Second	AASDT	False	
12	34	Analog_alarm_34	Internal AA23 switchDT: High limit violation dynamic		AADT-Internal	AASDT33	0	Second	AASDT	False	
13	35	Analog_alarm_35	Internal AA23 switchDT: delay 3 seconds High limit violation		AADT-Internal	AASDTdelay1	3	Second	AASDT	False	
14	42	Analog_alarm_40	Internal AA23 switchDT: delay 3 seconds Low limit violation LLV		AADT-Internal	AASDTdelay2	3	Second	AASDT	False	
15	23	Analog_alarm_41	AA23 DT in event text: <field ref="0" /> Bool, <field ref="1" /> Byte, <fi	<ref id = 0; type = AlarmTag; T	ACAT	HMI_TriggerAT	0	Second	AASDT	False	
16	24	Analog_alarm_42	AA24 DT in event text: <field ref="0" /> Timer, <field ref="1" /> Counte	<ref id = 0; type = AlarmTag; T	ACAT	HMI_TriggerAT	0	Second	AASDT	False	

Table 10- 13 Meaning of the entries

List entry	Meaning
ID	The alarm number is used to reference an alarm. The alarm number is unique. Alarms with identical alarm numbers are overwritten during import. An alarm without an existing alarm number is created.
Name	Name of the analog alarm
Event text [de-DE], Alarm text	Displays the alarm text. The field designation contains a language ID. Alarm texts must be assigned a language ID for import. An expression with a reference ID will be added to the alarm text if the text has a dynamic parameter. Example: text <field ref="0" />. Use the ID to assign dynamic parameters to alarm texts.
FeldInfo	Specifies whether the alarm text contains dynamic parameters. The settings are separated by a semicolon ";". Example of dynamic parameters: Tag: <ref id = 0; type = AlarmTag; Tag = Tag1; DisplayType = Decimal; Length = 5;> Text list: <ref id = 1; type = CommonTextList; TextList = Textlist1; Tag = tag 2; Length = 5;>

List entry	Meaning
Class	The class of an alarm determines whether or not the alarm must be acknowledged. It can also be used to determine how the alarm appears when it is displayed on the HMI device. The alarm class also determines whether and where the corresponding alarm will be logged.
Group	Indicates the allocation to an alarm group. If an alarm belongs to a group with other alarms, it can be acknowledged together with these alarms of the same group in a single operation.
Trigger tag	Specifies the tag monitored for limit value violation.
Delay time value	Specifies the delay time. The alarm is not triggered until the duration of the limit value violation equals the specified delay time.
Delay time unit	Specifies the time unit for the delay.
Report	Enables reporting of the specific alarm on a printer. True or "1" = Reporting enabled. False or "0" = Reporting disabled. Reporting must also be globally enabled in the project.
Info text [de-DE], Info text	The tooltip is an optional property of an alarm. Tooltips can contain additional information about the alarm. A tooltip will be displayed in a separate window on the HMI device when the operator presses the <HELP> key. The field designation contains a language ID.

Example of the worksheet "Limits"



	A	B	C	D	E	F	G
1	Alarm ID	Limit type	Limit value	Limit mode	Deadband mo	Deadband valu	Deadband in percent
2	1	Constant	0	Upper limit	Off	0	False
3	2	Constant	1	Upper limit	Off	0	False
4	3	Constant	2	Upper limit	Off	0	False
5	4	Constant	3	Upper limit	Off	0	False
6	5	Constant	4	Upper limit	Off	0	False
7	25	Constant	50	Upper limit	On both	5	False
8	26	Constant	50	Upper limit	On both	10	True
9	31	Constant	50	Lower limit	Off	0	False
10	32	Constant	50	Upper limit	Off	0	False
11	33	Tag	AASDTdyn	Lower limit	Off	0	False
12	34	Tag	AASDT1dyn	Upper limit	Off	0	False
13	35	Constant	50	Upper limit	Off	0	False
14	36	Constant	50	Lower limit	On both	5	False

Table 10- 14 Meaning of the entries

List entry	Meaning
Alarm ID	Alarm number The alarm number is used to reference an alarm. The alarm number is unique. Alarms with identical alarm numbers are overwritten during import. An alarm without an existing alarm number is created.
Limit mode	Trigger mode Indicates the method used for monitoring the limit value.
Limit type	Specifies the limit that will be monitored. Both a tag and a constant can be used as limit value.
Limit value	Limit value Indicates the tag or constant monitored for limit violation.
Deadband mode	Hysteresis mode Specifies whether and in which cases hysteresis will be used. For "Outgoing" For "Incoming" For "Incoming" and "Outgoing"
Deadband in percent	0 = The value specified for "Hysteresis" is considered to be absolute. 1 = The value specified for "Hysteresis" is referred to as a percentage of the limit value.
Deadband mode	Hysteresis Specifies a constant as a value of the hysteresis.

Note

"No value" in the table

Entries in the table which have the value "No value" delete the corresponding values in an existing alarm of the same name.

See also

Exporting alarms (Page 2698)

Format of the discrete alarm data

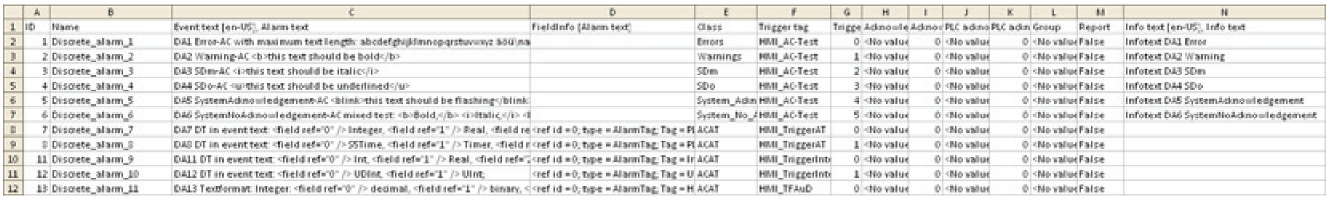
Introduction

This chapter describes the required format of the file for the import of discrete alarms. The file containing the discrete alarm data must be in "*.xlsx" format.

Structure of the alarm data

The import file in Microsoft Excel consists of the worksheets "Discrete alarms" (discrete alarms). Each alarm is assigned a separate row in the import file. Structure of the import file containing the discrete alarms:

Example of the worksheet "Discrete alarms"



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	ID	Name	Event text [en-US], Alarm text	FieldInfo [Alarm text]	Class	Trigger tag	Trigge	Acknowle	Acknov	PLC ackno	PLC ackn	Group	Report	Info text [en-US], Info text
2	1	Discrete_alarm_1	DA1 Error-AC with maximum text length: abcdefghijklmnopqrstuvwxyz äöü\ma		Errors	HMI_AC-Test	0	<No value	0	<No value	0	<No value	False	Infotext DA1 Error
3	2	Discrete_alarm_2	DA2 Warning-AC <b>this text should be bold</b>		Warnings	HMI_AC-Test	1	<No value	0	<No value	0	<No value	False	Infotext DA2 Warning
4	3	Discrete_alarm_3	DA3 SDm-AC <i>this text should be italic</i>		SDm	HMI_AC-Test	2	<No value	0	<No value	0	<No value	False	Infotext DA3 SDm
5	4	Discrete_alarm_4	DA4 SDo-AC <u>this text should be underlined</u>		SDo	HMI_AC-Test	3	<No value	0	<No value	0	<No value	False	Infotext DA4 SDo
6	5	Discrete_alarm_5	DA5 SystemAcknowledgement-AC <blink>this text should be flashing</blink		System_Ackn	HMI_AC-Test	4	<No value	0	<No value	0	<No value	False	Infotext DA5 SystemAcknowledgement
7	6	Discrete_alarm_6	DA6 SystemNoAcknowledgement-AC mixed text: <b>Bold,</b> <i>Italic,</i> <I		System_No_A	HMI_AC-Test	5	<No value	0	<No value	0	<No value	False	Infotext DA6 SystemNoAcknowledgement
8	7	Discrete_alarm_7	DA7 DT in event text: <field ref="0" /> Integer, <field ref="1" /> Real, <field re	<ref id = 0; type = AlarmTag; Tag = PL	ACAT	HMI_TriggerAT	0	<No value	0	<No value	0	<No value	False	
9	8	Discrete_alarm_8	DA8 DT in event text: <field ref="0" /> S5Time, <field ref="1" /> Timer, <field r	<ref id = 0; type = AlarmTag; Tag = PL	ACAT	HMI_TriggerAT	1	<No value	0	<No value	0	<No value	False	
10	11	Discrete_alarm_9	DA11 DT in event text: <field ref="0" /> Int, <field ref="1" /> Real, <field ref=",	<ref id = 0; type = AlarmTag; Tag = In	ACAT	HMI_TriggerInt	0	<No value	0	<No value	0	<No value	False	
11	12	Discrete_alarm_10	DA12 DT in event text: <field ref="0" /> UDInt, <field ref="1" /> UInt;	<ref id = 0; type = AlarmTag; Tag = U	ACAT	HMI_TriggerInt	1	<No value	0	<No value	0	<No value	False	
12	13	Discrete_alarm_11	DA13 Textformat: Integer: <field ref="0" /> decimal, <field ref="1" /> binary, <	<ref id = 0; type = AlarmTag; Tag = H	ACAT	HMI_TFAuD	0	<No value	0	<No value	0	<No value	False	

Table 10- 15 Meaning of the entries

List entry	Meaning
ID	The alarm number is used to reference an alarm. The alarm number is unique. Alarms with identical alarm numbers are overwritten during import. An alarm without an existing alarm number is created.
Name	Name of the analog alarm
Event text [de-DE], Alarm text	Displays the alarm text. The field designation contains a language ID. For import, a language ID must be assigned to alarm text. An expression with a reference ID will be added to the alarm text if the text has a dynamic parameter. Example: text <field ref="0" />. Use the ID to assign dynamic parameters to alarm texts.
FeldInfo	Specifies whether the alarm text contains dynamic parameters. The settings are separated by a semicolon ";". Example of dynamic parameters: Tag: <ref id = 0; type = AlarmTag; Tag = Tag1; DisplayType = Decimal; Length = 5;> Text list: <ref id = 1; type = CommonTextList; TextList = Textlist1; Tag = tag 2; Length = 5;>
Class	The class of an alarm determines whether or not the alarm must be acknowledged. It can also be used to determine how the alarm appears when it is displayed on the HMI device. The alarm class also determines whether and where the corresponding alarm will be logged.

List entry	Meaning
Group	Indicates the allocation to an alarm group. If an alarm belongs to a group with other alarms, it can be acknowledged together with these alarms of the same group in a single operation.
Trigger tag	Specifies the tag containing the bit that triggers the alarm.
Trigger bit	Specifies the number of the bit that triggers the alarm.
Acknowledge tag	Specifies the tag containing the bit that is set by the operator upon acknowledgment. Only available if the selected alarm class requires alarm acknowledgment.
Acknowledgment bit	Specifies the number of the bit that is set when the operator acknowledges the alarm.
PLC acknowledgement tag	Specifies the tag containing the bit that acknowledges the alarm of the control program. Only available if the selected alarm class requires alarm acknowledgment.
PLC acknowledgment bit	Specifies the number of the bit that acknowledges the alarm of the control program.
Delay time value	Specifies the delay time. The alarm is not triggered until the duration of the limit value violation equals the specified delay time.
Delay time unit	Specifies the time unit for the delay.
Report	Enables reporting of the specific alarm on a printer. True or "1" = Reporting enabled. False or "0" = Reporting disabled. Reporting must also be globally enabled in the project.
Info text [de-DE], Info text	The tooltip is an optional property of an alarm. Tooltips can contain additional information about the alarm. A tooltip will be displayed in a separate window on the HMI device when the operator presses the <HELP> key. The field designation contains a language ID.

Note

"No value" in the table

Entries in the table which have the value "No value" delete the corresponding values in an existing alarm of the same name.

See also

Exporting alarms (Page 2698)

10.9.2.4 Importing and exporting tags

Exporting tags

Introduction

WinCC has an export function for tags.

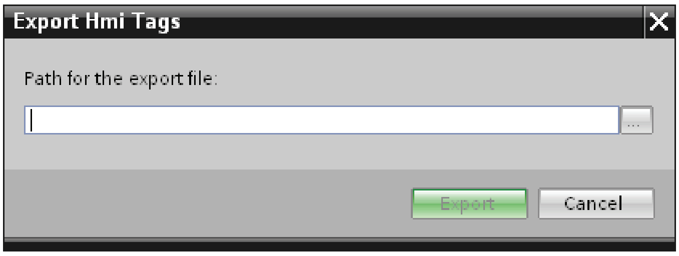
Requirements

- The WinCC project for the export is open.
- Tags have been created in the project.
- The "HMI tags" editor is open.

Exporting tags

1. Click on the ➡ button in the "HMI Tags" tab.

   The "Export" dialog box opens.



Export Hmi Tags

Path for the export file:

2. Click the "..." button and specify in which file the data are saved.
3. Click "Export". The export will start.

Result

The exported data has been written to an xlsx file. The xlsx file will be stored in the specified folder.

See also

Importing tags (Page 2707)

Exporting alarms (Page 2698)

Importing and exporting project data (Page 2693)

Exporting text lists (Page 2711)

Exporting recipes (Page 2694)

Format of the tag data (Page 2708)

Importing tags

Introduction

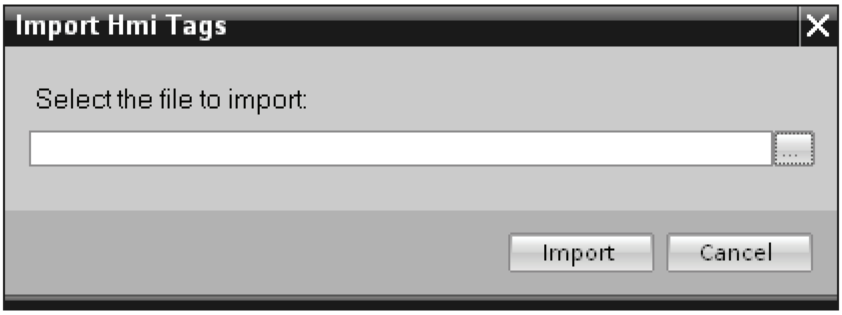
Tags are identified by the tag name. An existing tag will be overwritten with the data from the xlsx file if the tag name already exists in the project. A new tag is created if the tag does not yet exist.

Requirements

- An xlsx file with tags has been created.
- The structure of the xlsx file meets the requirements.
- The WinCC project for the import is open.
- The "HMI tags" editor is open.

Importing tags

1. Click on the button. The "Import" dialog box opens.



2. Click the "..." button and select the file that you want to import.
3. Click on the "Import" button. The import will start.

Result

The relevant tags have been created in WinCC. Alarms relating to the import operation are displayed in the output window. A log file is saved in the source directory of the import files. The log file has the same name as the respective import file but with the "*.xml" extension.

Check when importing the data whether there are any links to objects, for example, dynamic parameters such as tags.

- If an object with the same name already exists, the existing object is used.
- If no object of the same name yet exists, create an object with the relevant name or create a new link.

Note

The syntax of the import file is checked during xlsx file import. The meaning of the properties or dependencies between the properties is not checked. It is possible to assign a tag a trigger tag of the wrong type, for example, string. An error will be reported during compilation.

See also

Exporting tags (Page 2706)

Format of the tag data

Introduction

This section describes the format required for the file with tag data used for imports. The tag data file must be in "*.xlsx" format.

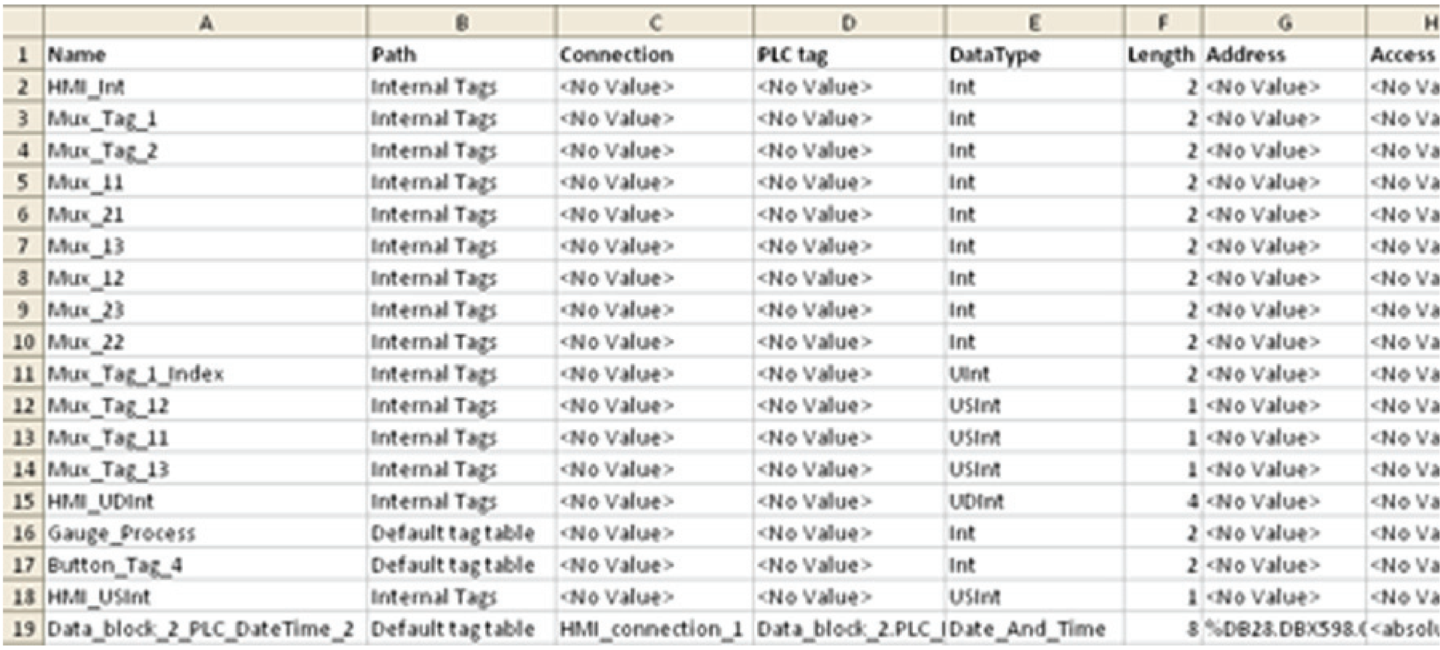
Tag data structure

The import file in Microsoft Excel consists of a number of worksheets:

- HMI Tags (HMI tags)
- Multiplexing (multiplex tags)

Each tag is on a separate line in the import file. The import file with the tag data must have the following format:

Example of the worksheet "HMI Tags"



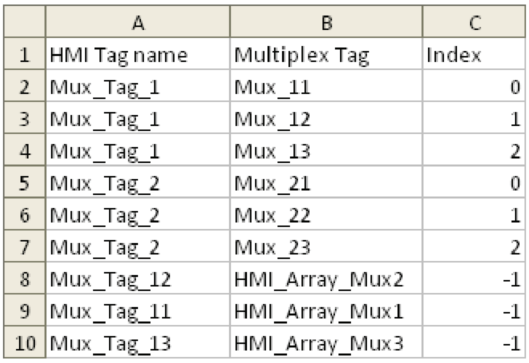
	A	B	C	D	E	F	G	H
1	Name	Path	Connection	PLC tag	DataType	Length	Address	Access
2	HMI_Int	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Va
3	Mux_Tag_1	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Va
4	Mux_Tag_2	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Va
5	Mux_11	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Va
6	Mux_21	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Va
7	Mux_13	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Va
8	Mux_12	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Va
9	Mux_23	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Va
10	Mux_22	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Va
11	Mux_Tag_1_Index	Internal Tags	<No Value>	<No Value>	UInt	2	<No Value>	<No Va
12	Mux_Tag_12	Internal Tags	<No Value>	<No Value>	USInt	1	<No Value>	<No Va
13	Mux_Tag_11	Internal Tags	<No Value>	<No Value>	USInt	1	<No Value>	<No Va
14	Mux_Tag_13	Internal Tags	<No Value>	<No Value>	USInt	1	<No Value>	<No Va
15	HMI_UDInt	Internal Tags	<No Value>	<No Value>	UDInt	4	<No Value>	<No Va
16	Gauge_Process	Default tag table	<No Value>	<No Value>	Int	2	<No Value>	<No Va
17	Button_Tag_4	Default tag table	<No Value>	<No Value>	Int	2	<No Value>	<No Va
18	HMI_USInt	Internal Tags	<No Value>	<No Value>	USInt	1	<No Value>	<No Va
19	Data_block_2_PLC_DateTime_2	Default tag table	HMI_connection_1	Data_block_2.PLC_I	Date_And_Time	8	%DB28.DBX598.(	<absolu

Table 10- 16 Meaning of the entries

List entry	Meaning
Name	Indicates the configured name of an HMI tag.
Path	Specifies which folders in the project tree contain the tag. The folder structure is represented by "\" : "FolderName1\FolderName2\TagName".
PLC Tag	Indicates whether the tag is linked to a PLC tag.
Connection	Indicates the name of the connection to the PLC.
Data type	Specifies the data type of a tag. The data types allowed depend on the communication driver being used. See the "Communication" section of the documentation for additional information on the data types permitted for the various communication drivers.
Length	Specifies the length of the tag. This entry is only useful for data types with a dynamic length such as strings; it is left empty for all other data types.
Address	Specifies the tag address in the PLC. The tag address must exactly match the one used in WinCC, for example, "%DB1.DBW0". The tag address is empty for internal tags.
Multiplexing	Specifies whether multiplexing is used.
Index tag	Shows the name of the index tag for multiplexing. In Runtime, the system first reads the value of the index tag. It then accesses the tag in the corresponding place in the tag list.
StartValue	Specifies the start value of a tag.
ID tag	The update ID updates the value of a tag with the aid of a function or a PLC job. The update ID must be unique within an HMI device.
Coding	Shows the coding method.

List entry	Meaning
DiplayName [de_DE]	Shows the display name of an HMI tag. The field designation contains a language ID. The field designation contains a language ID. Display names must be assigned a language ID for import. Texts are imported to the corresponding project language.
Acquisition mode	Specifies the tag acquisition mode.
Acquisition cycle	Specifies the tag acquisition cycle. The acquisition cycle must correspond exactly to the one used in WinCC. The value is not language-dependent and should therefore be the same in every language. The default value is "1 s". The acquisition cycle is undefined if the tag acquisition mode is "on demand". User-defined acquisition cycles must be created beforehand as the file will otherwise not be imported.
High High Limit type	Indicates whether the limit value "High high" is monitored by a constant, a tag or not at all.
High High Limit	Displays the limit value "High High".
High Limit type	Indicates whether the limit value "High" is monitored by a constant, a tag or not at all.
High Limit	Displays the limit value "High".
Low Limit type	Indicates whether the limit value "Low" is monitored by a constant, a tag or not at all.
Low Limit	Displays the limit value "Low".
Low Low Limit type	Indicates whether the limit value "Low Low" is monitored by a constant, a tag or not at all.
Low Low Limit	Displays the limit value "Low Low".
Linear scaling	Indicates whether linear scaling is enabled. This entry can only be used for external tags.
End value PLC	Specifies the end value of the PLC tag.
Start value PLC	Specifies the start value of the PLC tag.
End value HMI	Specifies the end value of the HMI tag.
Start value HMI	Specifies the start value of the HMI tag.

Example of the worksheet "Multiplexing"



	A	B	C
1	HMI Tag name	Multiplex Tag	Index
2	Mux_Tag_1	Mux_11	0
3	Mux_Tag_1	Mux_12	1
4	Mux_Tag_1	Mux_13	2
5	Mux_Tag_2	Mux_21	0
6	Mux_Tag_2	Mux_22	1
7	Mux_Tag_2	Mux_23	2
8	Mux_Tag_12	HMI_Array_Mux2	-1
9	Mux_Tag_11	HMI_Array_Mux1	-1
10	Mux_Tag_13	HMI_Array_Mux3	-1

Table 10- 17 Meaning of the entries

List entry	Meaning
Name	Indicates the configured name of an HMI tag which uses indirect addressing. The HMI tag must be available in the "HMI Tags" worksheet.
Index	Shows the value which governs which tag is selected.
Multiplex Tag	Displays the tag from the tag list corresponding to the index value.

Note

"No value" in the table

Entries in the table which have the value "No value" delete the corresponding values in an existing tag of the same name.

See also

Exporting tags (Page 2706)

10.9.2.5 Importing and exporting text lists

Exporting text lists

Introduction

WinCC has an export function for text lists.

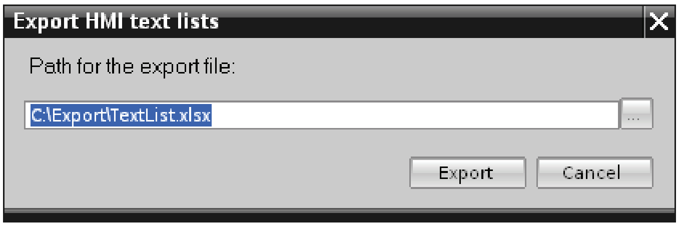
Requirements

- The WinCC project for the export is open.
- Text lists have been created in the project.
- The "Text & graphics lists" editor is open.

Exporting text lists

1. Click on the button in the "TextLists" tab.

   The "Export" dialog box opens.



Export HMI text lists

Path for the export file:

C:\Export\TextList.xlsx

2. Click the "..." button and specify in which file the data are saved.
3. Click "Export". The export will start.

Result

The exported data has been written to an xlsx file. The xlsx file will be stored in the specified folder.

See also

Importing text lists (Page 2712)

Importing and exporting project data (Page 2693)

Exporting tags (Page 2706)

Exporting recipes (Page 2694)

Format of text list data (Page 2713)

Importing text lists

Introduction

You then import text lists from an xlsx file to WinCC.

Requirements

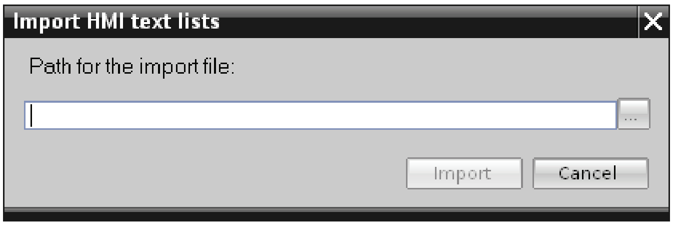
- An xlsx file with text lists has been created.
- The structure of the xlsx file meets the requirements.

- The WinCC project for the import is open.
- The "Text & graphics lists" editor is open.

Importing text lists

1. Click on the button in the "Text lists" tab.

   The "Import" dialog box opens.

2. Select the file you want to import under "File selection".
3. Click on the "Import" button. The import will start.

Result

You have now imported the text lists. The relevant text lists have been created in WinCC. Alarms relating to the import operation are displayed in the output window. A log file is saved in the source directory of the import files. The log file has the same name as the respective import file but with the "*.xml" extension.

Check when importing the data whether there are any links to objects, for example, dynamic parameters such as tags.

- If an object with the same name already exists, the existing object is used.
- If no object of the same name yet exists, create an object with the relevant name or create a new link.

See also

Exporting text lists (Page 2711)

Format of text list data

Introduction

This section describes the format required for the file with the text lists used for imports. The text list data file must be in "*.xlsx" format.

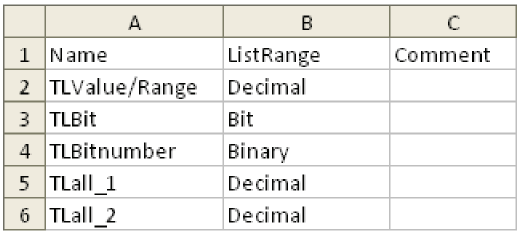
Tag data structure

The import file in Microsoft Excel consists of two worksheets:

- TextList (Text lists)
- TextListEntry (Text list entry)

Each text list is assigned a separate line in the import file. The import file containing the data must be structured as follows:

Example of the worksheet "TextList"



	A	B	C
1	Name	ListRange	Comment
2	TLValue/Range	Decimal	
3	TLBit	Bit	
4	TLBitnumber	Binary	
5	TLall_1	Decimal	
6	TLall_2	Decimal	

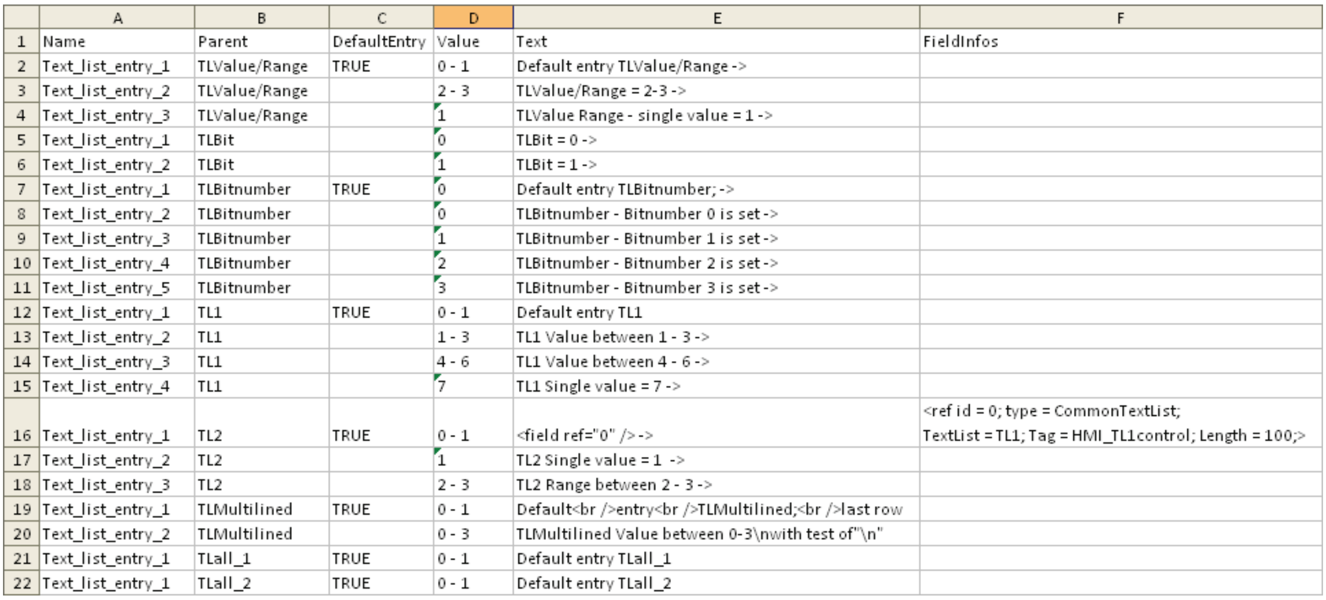
Table 10- 18 Meaning of the entries

List entry	Meaning
Name	Shows the name of the text list.

ListRange	Shows the text list range:
	Number = Bit number (0-31)
	Range = value/range (...-...)
	Bit = Bit (0;1)

Comment	Any comments on the text list. You can use up to 500 characters

Example of the worksheet "TextListEntry"



	A	B	C	D	E	F
1	Name	Parent	DefaultEntry	Value	Text	FieldInfos
2	Text_list_entry_1	TLValue/Range	TRUE	0 - 1	Default entry TLValue/Range ->	
3	Text_list_entry_2	TLValue/Range		2 - 3	TLValue/Range = 2-3 ->	
4	Text_list_entry_3	TLValue/Range		1	TLValue Range - single value = 1 ->	
5	Text_list_entry_1	TLBit		0	TLBit = 0 ->	
6	Text_list_entry_2	TLBit		1	TLBit = 1 ->	
7	Text_list_entry_1	TLBitnumber	TRUE	0	Default entry TLBitnumber; ->	
8	Text_list_entry_2	TLBitnumber		0	TLBitnumber - Bitnumber 0 is set ->	
9	Text_list_entry_3	TLBitnumber		1	TLBitnumber - Bitnumber 1 is set ->	
10	Text_list_entry_4	TLBitnumber		2	TLBitnumber - Bitnumber 2 is set ->	
11	Text_list_entry_5	TLBitnumber		3	TLBitnumber - Bitnumber 3 is set ->	
12	Text_list_entry_1	TL1	TRUE	0 - 1	Default entry TL1	
13	Text_list_entry_2	TL1		1 - 3	TL1 Value between 1 - 3 ->	
14	Text_list_entry_3	TL1		4 - 6	TL1 Value between 4 - 6 ->	
15	Text_list_entry_4	TL1		7	TL1 Single value = 7 ->	
16	Text_list_entry_1	TL2	TRUE	0 - 1	<field ref="0" /> ->	<ref id = 0; type = CommonTextList; TextList = TL1; Tag = HMI_TL1control; Length = 100;>
17	Text_list_entry_2	TL2		1	TL2 Single value = 1 ->	
18	Text_list_entry_3	TL2		2 - 3	TL2 Range between 2 - 3 ->	
19	Text_list_entry_1	TLMultilined	TRUE	0 - 1	Default<br />entry<br />TLMultilined;<br />last row	
20	Text_list_entry_2	TLMultilined		0 - 3	TLMultilined Value between 0-3\nwith test of"\n"	
21	Text_list_entry_1	TLall_1	TRUE	0 - 1	Default entry TLall_1	
22	Text_list_entry_1	TLall_2	TRUE	0 - 1	Default entry TLall_2	

Table 10- 19 Meaning of the entries

List entry	Meaning
Name	Shows the name of the text list entry.
Parent	Specifies the name of the corresponding text list.
DefaultEntry	Indicates whether the text list entry is a default entry. The default entry is always displayed when the tag has an undefined value.
Value	Specifies the tag integer values or value ranges which are assigned to the text entries in the text list.
Text	Shows the text list entry. The field designation contains a language ID. Text list entries must be assigned a language ID for import. An expression with a reference ID will be added to the text if the text list entry has a dynamic parameter. Example: text <field ref="0" />. Use the ID to assign the dynamic parameter to a text list entry.
FeldInfo	Specifies whether the text list contains dynamic parameters. The settings are separated by a semicolon ";". Example of dynamic parameters: Tag: <ref id = 0; type = CommonTagDisplayFormat; Tag = tag 1; DisplayType = Decimal; DisplayFormat = 9;> Text list: <ref id = 1; type = CommonTextList; TextList = Textliste_1; Tag = tag 2; Length = 5;> PLC tag: <ref id = 0; type = CommonControlTagDisplayFormat; DisplayType = Decimal; DisplayFormat = 9;>

See also

Exporting text lists (Page 2711)

10.9.2.6 Importing and exporting project texts

Exporting project texts

Project texts are exported for translation. Texts are exported to Office Open XML files ending in ".xlsx". These files can be edited in Microsoft Excel, for example.

You can exchange the file with the translators and import it back to the project as soon as it has been translated.

Requirements

- At least two languages have been enabled in the "Project languages" editor, for example, Italian and French.

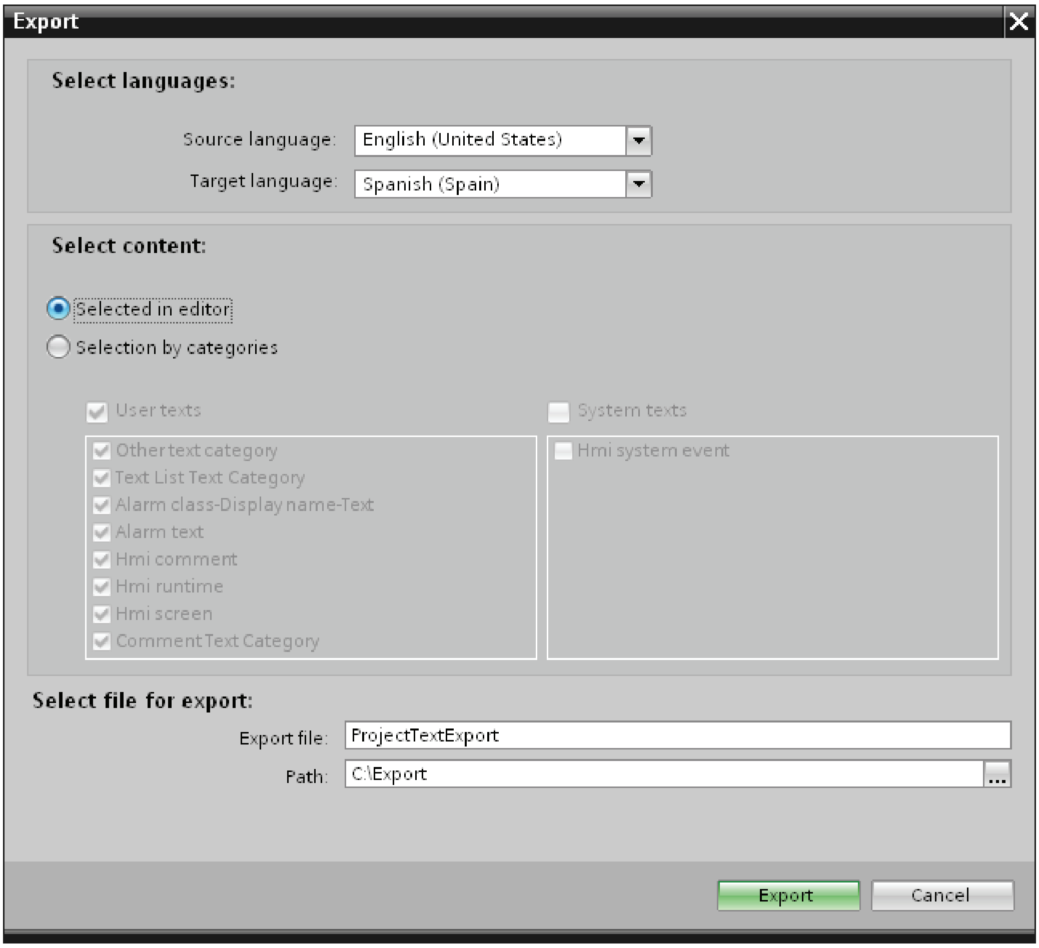
Exporting project texts

To export individual project texts, proceed as follows:

1. Click on the arrow to the left of "Languages & resources" in the project tree.

   The child elements are displayed.
2. Double-click on "Project texts". The "Project texts" editor will open.
3. Select the texts you want to export.

4. Click on the ⇨ button. The "Export" dialog opens.



Export

Select languages:

Source language: English (United States)

Target language: Spanish (Spain)

Select content:

● Selected in editor

○ Selection by categories

☑ User texts

- ☑ Other text category
- ☑ Text List Text Category
- ☑ Alarm class-Display name-Text
- ☑ Alarm text
- ☑ Hmi comment
- ☑ Hmi runtime
- ☑ Hmi screen
- ☑ Comment Text Category

□ System texts

- □ Hmi system event

Select file for export:

Export file: ProjectTextExport

Path: C:\Export

5. From the "Source language" drop-down list, select the language from which you wish to translate, for example Italian.
6. From the "Target language" drop-down list, select the language into which the texts are to be translated, for example, French.
7. Enter a file path and a file name for the export file in the "Export file" input field.
8. Click "Export".

Result

The texts selected in the "Project texts" editor are written to an xlsx file. The xlsx file will be stored in the specified folder.

You can alternatively select and export all project texts from categories. Select "User texts" or "System texts" in the "Export" dialog in line with the type of texts you wish to export. In this case, export can additionally be limited by categories.

Note

Project texts in library objects cannot be exported.

See also

Importing project texts (Page 2718)

Importing project texts

Edit the xlsx file or send it to a translator. Import the texts once they have been translated. The foreign languages will be imported to the relevant object in the project.

Requirements

- At least two languages have been enabled in the "Project languages" editor, for example, Italian and French.

Importing project texts

To import a project text file, proceed as follows:

1. Click on the arrow to the left of "Languages & resources" in the project tree.

   The lower-level elements will be displayed.
2. Double-click on "Project texts". The "Project texts" editor will open.
3. Click on the ← button. The "Import" dialog opens.
4. Select the path and file name of the import file from the "Import file" field.
5. Activate the "Import source language" check box if you have made changes to the source language in the export file and would like to overwrite the entries in the project with the changes.
6. Click on "Import".

Result

You have imported the project texts.

See also

Exporting project texts (Page 2716)

10.9.3 Using cross-references

10.9.3.1 General information about cross references

Introduction

The cross-reference list provides an overview of the use of objects within the project.

Uses of cross-references

The cross-reference list offers you the following advantages:

- When creating and changing a program, you retain an overview of the objects, tags, and alarms etc. you have used.
- From the cross-references, you can jump directly to the object location of use.
- You can learn the following when debugging:
  - The objects used in a specific screen.
  - The alarms and recipes shown in a specific display.
  - The tags used in a specific alarm or object.
- As part of the project documentation, the cross-references provide a comprehensive overview of all object, alarms, recipes, tags and screens.

10.9.3.2 Displaying the cross-reference list

Introduction

Details on the use of objects can be found in the cross-reference list. You can show cross-references for HMI devices, folders and all editors in the project tree. The detail view also lets you select individual objects of the editors.

Requirement

You have created a project.

Several objects have been created.

Procedure

1. Select the required entry in the project tree or detail view.
2. Select "Cross-references" in the shortcut menu. The cross-reference list is opened in the work area.
3. Open the "Used by" tab to display where the objects shown in the cross-reference list are used.
4. Open the "Used" tab to view the users of the objects displayed in the cross-reference list.
5. You can sort the entries in the "Object" column in ascending or descending order by clicking on the corresponding column header.
6. To go to the location of use for a specific object, click on the displayed link.

Result

The cross-reference list for the selected object is displayed in the work area.

10.9.3.3 Structure of the cross-reference list

Views of the cross-reference list

There are two views of the cross-reference list. The difference between the two views is in the objects displayed in the first column:

- Used by:

  Display of the referenced objects. Here, you can see where the object is used.

- Used:

  Display of the referencing objects. The users of the object are shown here.

The assigned tool tips provide additional information about each object.

Structure of the cross-reference list

Column	Content/meaning
Object	Name of the object that uses the lower-level objects or that is being used by the lower-level objects.
Numbers	Number of uses
Location of use	Each location of use, for example, an object or event
Property	Function of the referenced objects, for example, tag for data record or process value
Connected to	PLC tag with which the object is connected.
Type	Type of object
Path	Path of object

Depending on the installed products, additional columns or different columns are displayed for the cross-references.

Settings in the cross-reference list

You can make the following settings using the icons in the toolbar for the cross-reference list:

- Update cross-reference list

  Updates the current cross-reference list.

- Making settings for the cross-reference list

  Here, you specify whether all used, all unused, all defined or all undefined objects will be displayed. If the "Undefined objects" option is enabled, references to previously deleted objects are also displayed.

- Collapse entries

  Reduces the entries in the current cross-reference list by closing the lower-level objects.

- Expand entries

  Expands the entries in the current cross-reference list by opening the low-level objects.

Sorting in the cross-reference list

You can sort the entries in the "Object" column in ascending or descending order. Click on the column header to do this.

10.9.3.4 Displaying cross-references in the Inspector window

Introduction

The Inspector window displays cross-reference information about a selected object in the "Info > Cross-references" tab. The Inspector window displays the cross-reference information in tabular format.

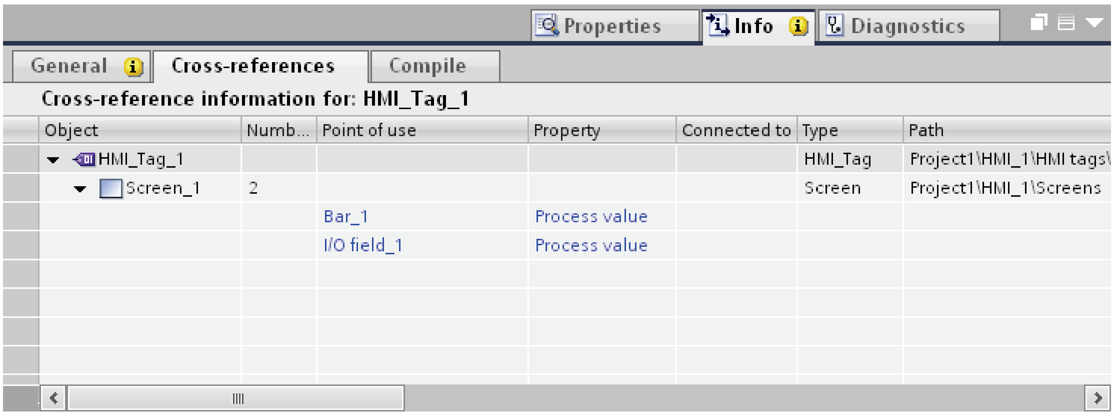
Requirement

You have created a project.

Several objects have been created.

Procedure

1. Select an object in a screen or a tabular editor.
2. Select "Cross-reference information" in the shortcut menu. The cross-references are opened in the Inspector window.



Cross-reference information for: HMI_Tag_1

Object	Numb...	Point of use	Property	Connected to	Type	Path
HMI_Tag_1					HMI_Tag	Project1\HMI_1\HMI tags\
Screen_1	2				Screen	Project1\HMI_1\Screens
		Bar_1	Process value			
		I/O field_1	Process value			

Result

The instances where and the other objects by which the selected object is being used are displayed.

The table below shows the additional information listed in the "About > Cross-reference" tab:

Column	Meaning
Object	Name of the object that uses the lower-level objects or that is being used by the lower-level objects.
Numbers	Number of uses
Location of use	Each location of use, for example an object or event
Property	Function of the referenced objects, for example tag for data record or process value
Connected to	PLC tag with which the object is connected.
Type	Type of object
Path	Path of object

Depending on the installed products, additional columns or different columns are displayed for the cross-references.

10.9.4 Managing languages

10.9.4.1 Languages in WinCC

User interface language and project languages

A distinction is drawn between two different language levels in WinCC:

- User interface language

  During configuration, the text in the WinCC menus and dialogs is displayed in the user interface language. The user interface language also affects the labeling of operating elements, the parameters of the system functions, the online help, etc.

- Project languages

  Project languages are all languages in which a project will later be used. Project languages are used to create a project in multiple languages.

The two language levels are completely independent of one another. For example, you can create English projects at any time using a German user interface and vice versa.

Project languages

The following languages are differentiated within the project languages:

- Reference language

  The reference language is the language that you use to configure the project initially.

  During configuration, you select one of the project languages as the reference language. You use the reference language as a template for translations. All of the texts for the project are first created in the reference language and then translated. While you are translating the texts, you can have them displayed simultaneously in the reference language.

- Editing language

  You produce translations of the texts in the editing language.

  Once you have created your project in the reference language, you can translate the texts into the remaining project languages. Select a project language respectively as an edit language and edit the texts for the appropriate language variant. You can change the editing language at any time.

Note

When switching the project languages, the assignment to the keys on the keyboard also changes. For some languages (for example, Spanish), the operating system does not allow you to switch to the corresponding keyboard assignment. In this case, the keyboard assignment is switched to English.

- Runtime languages

  Runtime languages are those project languages that are transferred to the HMI device. You decide which project languages to transfer to the HMI device depending on your project requirements.

  You must provide appropriate controls so that the operator can switch between languages in runtime.

See also

Language settings in the operating system (Page 2724)

Operating system settings for Asian languages (Page 2725)

Selecting the user interface language (Page 2726)

"Graphics" editor (Page 2737)

Languages in Runtime (Page 2741)

Example: Configuring a button for language switching (Page 2748)

10.9.4.2 Language settings in the operating system

Introduction

The configuration PC operating system settings influence WinCC language management in the following areas:

- Selection of project languages
- Regional format of dates, times, currency, and numbers
- Displaying ASCII characters

Project language selection

A language is not available as a project language unless it is installed in the operating system.

Regional format of dates, times, currency, and numbers

WinCC specifies a fixed date and time format in the Date - Time field for the selected project language and runtime language.

In order for dates, times, and numbers to be presented correctly in the selected editing language, this language must be set in the Regional Options in the Control Panel.

Displaying ASCII characters

With text output fields, the display of ASCII characters as of 128 depends on the set language and the operating system being used.

If the same special characters are to be displayed on different PCs, the PCs must use the same operating system and regional settings.

See also

Languages in WinCC (Page 2723)

Operating system settings for Asian languages (Page 2725)

10.9.4.3 Operating system settings for Asian languages

Settings on Western operating systems

If you want to enter Asian characters, you must activate the support for this language in the operating system.

The Input Method Editor (IME) is available in Windows for configuring Asian texts. Without this editor, you can display Asian text but not edit it. For more information on the Input Method Editor, refer to the documentation for Windows. To enter Asian characters when configuring, switch to the Asian entry method in the "Input Method Editor".

Switch the operating system to the appropriate language to have language-specific project texts, such as alarm texts, displayed in the simulator in Asian characters.

Settings on Asian operating systems

If you are configuring on an Asian operating system, you must switch to the English default input language to enter ASCII characters, for example, for object names. As the English default input language is included in the basic installation of the operating system, you do not need to install an additional input locale.

Enable language support

1. Open the system controller.
2. Select "Regional and Language Options".
3. On the "Languages" tab, activate the check box "Install files for East Asian languages".
4. Then click on "Details" under "Text Services and Input Languages". The dialog "Text Services and Input Languages" is opened.
5. On the "Settings" tab add the required default input language under the "Installed Services".
6. Select the language of the operating system in the "Language for non-Unicode programs" area in the "Advanced" tab.

See also

Languages in WinCC (Page 2723)

Language settings in the operating system (Page 2724)

10.9.4.4 Setting project languages

Selecting the user interface language

Introduction

The user interface language is used for displaying menu entries, title bars, infotexts, dialog texts and other designations in the WinCC user interface.

You can switch between the installed user interface languages during configuration. The labeling of the operating elements remains in the language you set when you added the object even if you change the user interface language.

Procedure

1. Select "Options > Settings" in the menu.

   The "Settings" dialog box is opened.
2. Select the desired user interface language under "General > General settings".

Result

WinCC will use the selected language as user interface language.

See also

Enable project languages (Page 2727)

Selecting the reference language and editing language (Page 2727)

Languages in WinCC (Page 2723)

Enable project languages

Introduction

The project languages are set in the "Project languages" editor. You define which project language is to be the reference language and which the editing language.

Enable project languages

1. Click on the arrow to the left of "Languages & resources" in the project tree.

   The lower-level elements will be displayed.
2. Double-click on "Project languages".

   The possible project languages will be displayed in the working area.
3. Enable the relevant project languages.

Note

Copying multilingual objects

The copies of multilingual objects to a different project only include text objects in the project languages which are activated in the target project. Activate all project languages in the target project to include the corresponding text objects when transferring the copy.

Disabling project languages

1. Disable the languages which are not relevant for the project.

CAUTION
If you disable a project language, all text and graphic objects you have already created in this language will be deleted from the current project.

See also

Selecting the user interface language (Page 2726)

Selecting the reference language and editing language (Page 2727)

Selecting the reference language and editing language

Introduction

The project languages are set in the "Project languages" editor. You define which project language is to be the reference language and which the editing language. You can change the editing language at any time.

Requirements

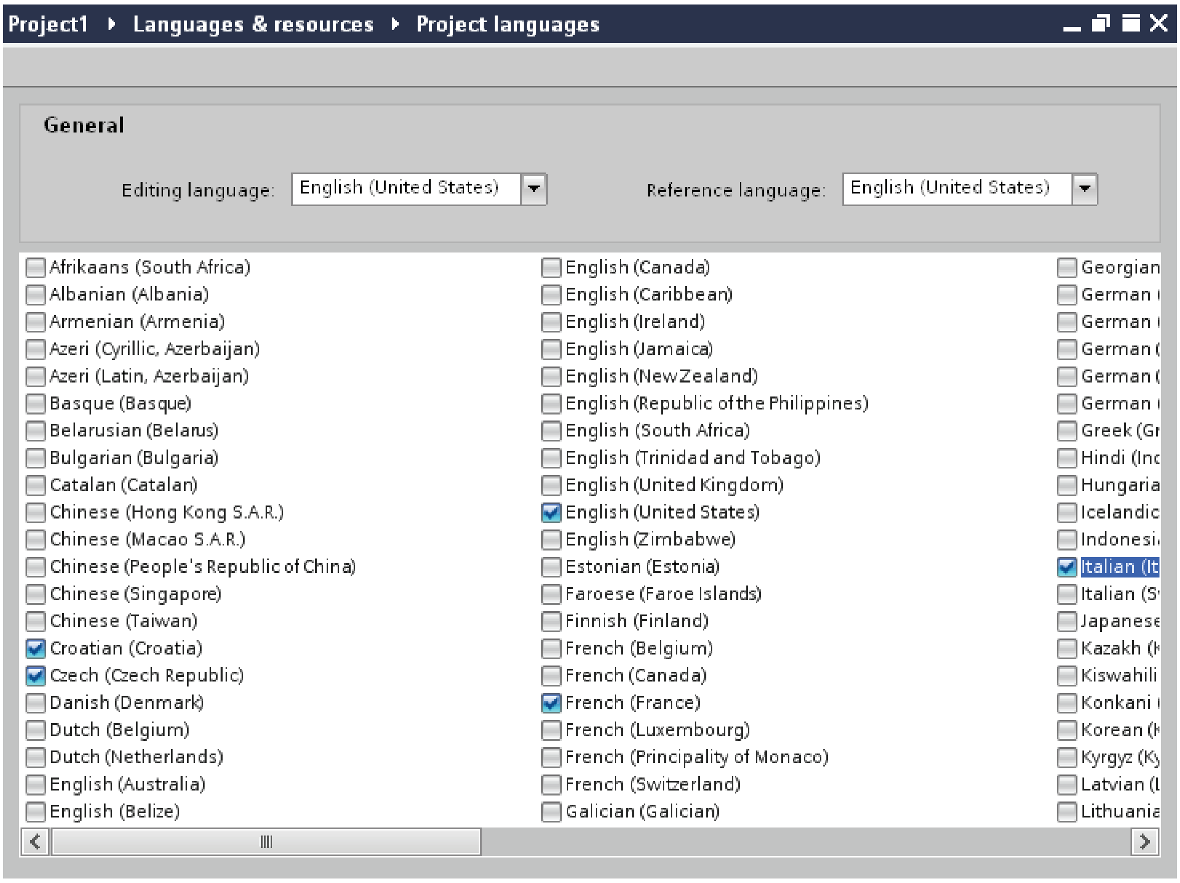
The "Project languages" editor is open.

Several project languages have been activated.

Selecting the reference language and editing language

1. Click the arrow in the drop-down list in the "General > Editing language" section.
2. Click on the required language in the drop-down list, for example, German.
3. Click on the arrow in the drop-down list in the "General > Reference language" section.
4. Click on the required language in the drop-down list, for example, English.

The language selection is displayed in the list box.



Project1 ▸ Languages & resources ▸ Project languages

General

Editing language: English (United States)

Reference language: English (United States)

☐ Afrikaans (South Africa)
☐ Albanian (Albania)
☐ Armenian (Armenia)
☐ Azeri (Cyrillic, Azerbaijan)
☐ Azeri (Latin, Azerbaijan)
☐ Basque (Basque)
☐ Belarusian (Belarus)
☐ Bulgarian (Bulgaria)
☐ Catalan (Catalan)
☐ Chinese (Hong Kong S.A.R.)
☐ Chinese (Macao S.A.R.)
☐ Chinese (People's Republic of China)
☐ Chinese (Singapore)
☐ Chinese (Taiwan)
☑ Croatian (Croatia)
☑ Czech (Czech Republic)
☐ Danish (Denmark)
☐ Dutch (Belgium)
☐ Dutch (Netherlands)
☐ English (Australia)
☐ English (Belize)

☐ English (Canada)
☐ English (Caribbean)
☐ English (Ireland)
☐ English (Jamaica)
☐ English (New Zealand)
☐ English (Republic of the Philippines)
☐ English (South Africa)
☐ English (Trinidad and Tobago)
☐ English (United Kingdom)
☑ English (United States)
☐ English (Zimbabwe)
☐ Estonian (Estonia)
☐ Faroese (Faroe Islands)
☐ Finnish (Finland)
☐ French (Belgium)
☐ French (Canada)
☑ French (France)
☐ French (Luxembourg)
☐ French (Principality of Monaco)
☐ French (Switzerland)
☐ Galician (Galician)

☐ Georgian
☐ German (
☐ German (
☐ German (
☐ German (
☐ German (
☐ Greek (Gr
☐ Hindi (Inc
☐ Hungaria
☐ Icelandic
☐ Indonesi
☑ Italian (It
☐ Italian (S
☐ Japanese
☐ Kazakh (K
☐ Kiswahili
☐ Konkani (
☐ Korean (K
☐ Kyrgyz (Ky
☐ Latvian (L
☐ Lithuania

Result

You have now selected the editing and reference languages.

If you change the editing language, all future text input will be stored in the new editing language.

See also

Selecting the user interface language (Page 2726)

Enable project languages (Page 2727)

10.9.4.5 Creating one project in multiple languages

Working with multiple languages

Multilingual configuration in WinCC

You can configure your projects in multiple languages using WinCC. There are various grounds for creating a project in multiple languages:

- You would like to use a project in more than one country.

  You create the project in multiple languages but when the HMI device is commissioned, only the language spoken by the operators at the respective site will be transferred to the HMI device.

- The operators of a system speak a range of different languages.

  Example: An HMI device is used in China, but the service personnel understand only English.

Translating project texts

With WinCC, you can enter project texts directly in several languages in various different editors, for example, in the "Project texts" editor. WinCC also allows you to export and import your configuration for translation purposes. This is particularly advantageous if you configure projects containing a large amount of text and want to have it translated.

Language management and translation in WinCC

The following editors are used to manage languages and translate texts in WinCC:

Editor	Short description
Project languages	Selection of project languages, editing language and reference language.
Languages and fonts	Management of runtime languages and fonts used on the HMI device.
Project texts	Central management of configured texts in all project languages.
Graphics	Graphic library for managing graphics and their language-specific versions.

See also

Project text basics (Page 2730)

Translating texts directly (Page 2731)

Translating texts using reference texts (Page 2733)

Exporting project texts (Page 2734)

Importing project texts (Page 2736)

Project text basics

Texts in different languages in the project

Texts that are output on display devices during processing are typically entered in the language in which the automation solution is programmed. Comments and the names of objects are also entered in this language.

If operators do not understand this language, they require a translation of all operator-relevant texts into a language they understand. You can therefore translate all the texts into any language. In this way, you can ensure that anyone who is subsequently confronted with the texts in the project sees the texts in his/her language of choice.

User texts and system texts

In the interests of clarity, a distinction is drawn between user texts and system texts:

- User texts are texts created by the user.
- System texts are texts created automatically and which are a product of configuration in the project.

The project texts are managed in the project text editor. This can be found in the project tree under "Languages & Resources > Project texts".

Examples of multilingual project texts

You can, for example, manage the following types of text in more than one language:

- Display texts
- Alarm texts
- Comments in tables
- Labels of screen objects
- Text lists

Translating texts

There are two ways of translating texts.

- Translating texts directly

  You can enter the translations for the individual project languages directly in the "Project texts" editor.
- Translating texts using reference texts

  You can change the editing language for shorter texts. You can enter the new texts in the editing language while the texts of the reference language are displayed.

See also

Working with multiple languages (Page 2729)

Translating texts directly (Page 2731)

Translating texts using reference texts (Page 2733)

Exporting project texts (Page 2734)

Importing project texts (Page 2736)

Translating texts directly

Translating texts

If you use several languages in your project, you can translate individual texts directly. As soon as you change the language of the software user interface, the translated texts are available in the selected language.

Requirements

- You are in the project view.
- A project is open.
- You have selected at least two further project languages.

Procedure

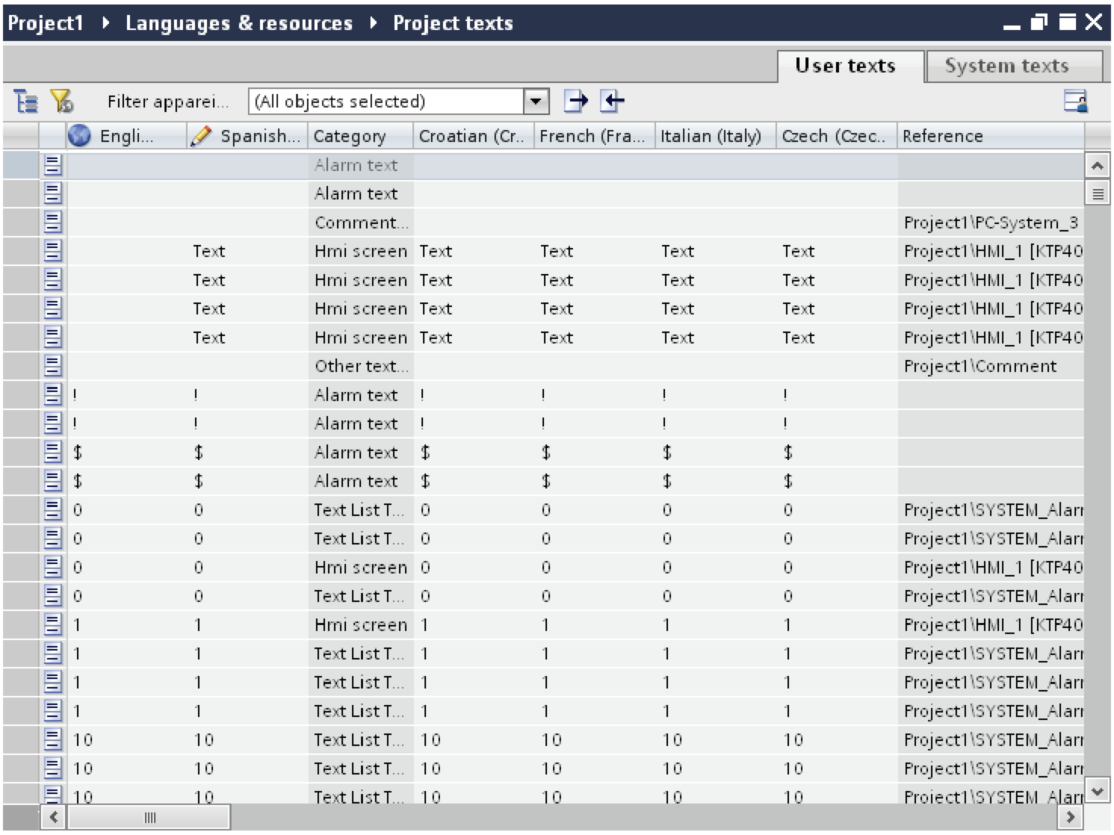
Proceed as follows to translate individual texts:

1. Click on the arrow to the left of "Languages & resources" in the project tree.

   The elements below this are displayed.

2. Double-click on "Project texts".

   A list with the texts in the project is displayed in the work area. There is a separate column for each project language.



Engli...	Spanish...	Category	Croatian (Cr..	French (Fra...	Italian (Italy)	Czech (Czec..	Reference
		Alarm text					
		Alarm text					
		Comment...					Project1\PC-System_3
	Text	Hmi screen	Text	Text	Text	Text	Project1\HMI_1 [KTP40
	Text	Hmi screen	Text	Text	Text	Text	Project1\HMI_1 [KTP40
	Text	Hmi screen	Text	Text	Text	Text	Project1\HMI_1 [KTP40
	Text	Hmi screen	Text	Text	Text	Text	Project1\HMI_1 [KTP40
		Other text...					Project1\Comment
!	!	Alarm text	!	!	!	!	
!	!	Alarm text	!	!	!	!	
$	$	Alarm text	$	$	$	$	
$	$	Alarm text	$	$	$	$	
0	0	Text List T...	0	0	0	0	Project1\SYSTEM_Aları
0	0	Text List T...	0	0	0	0	Project1\SYSTEM_Aları
0	0	Hmi screen	0	0	0	0	Project1\HMI_1 [KTP40
0	0	Text List T...	0	0	0	0	Project1\SYSTEM_Aları
1	1	Hmi screen	1	1	1	1	Project1\HMI_1 [KTP40
1	1	Text List T...	1	1	1	1	Project1\SYSTEM_Aları
1	1	Text List T...	1	1	1	1	Project1\SYSTEM_Aları
1	1	Text List T...	1	1	1	1	Project1\SYSTEM_Aları
10	10	Text List T...	10	10	10	10	Project1\SYSTEM_Aları
10	10	Text List T...	10	10	10	10	Project1\SYSTEM_Aları
10	10	Text List T...	10	10	10	10	Project1\SYSTEM_Aları

3. To group identical texts and translate them simultaneously, click on the "☰" button in the toolbar.

4. To hide texts that do not have a translation, click on the button in the toolbar.

5. Click on an empty column and enter the translation.

Result

You have translated individual texts in the "Project texts" editor. The texts will then be displayed in the runtime language.

See also

Working with multiple languages (Page 2729)

Exporting project texts (Page 2734)

Project text basics (Page 2730)

Importing project texts (Page 2736)

Translating texts using reference texts

Introduction

After changing the editing language, all texts are shown in input boxes in the new editing language. If there is not yet a translation available for this language, the input boxes are empty or filled with default values.

If you enter text again in an input field, this is saved in the current editing language. Following this, the texts exist in two project languages for this input field, in the previous editing language and in the current editing language. This makes it possible to create texts in several project languages.

You can display existing translations for an input box in other project languages. These serve as a comparison for text input in the current editing language and they are known as the reference language.

Requirement

There is at least one translation into a different project language for an input field.

Procedure

To display the translation of an input cell in a reference language, follow these steps:

1. Select "Tasks > Languages & resources" in the task card.
2. Select a reference language from the "Reference language" drop-down list.

Result

The reference language is preset. If you click in a text block, translations that already exist in other project languages are shown in the "Tasks > Reference text" task card.

See also

Working with multiple languages (Page 2729)

Exporting project texts (Page 2734)

Project text basics (Page 2730)

Importing project texts (Page 2736)

Exporting project texts

Project texts are exported for translation. Texts are exported to Office Open XML files ending in ".xlsx". These files can be edited in Microsoft Excel, for example.

You can exchange the file with the translators and import it back to the project as soon as it has been translated.

Requirements

- At least two languages have been enabled in the "Project languages" editor, for example, Italian and French.

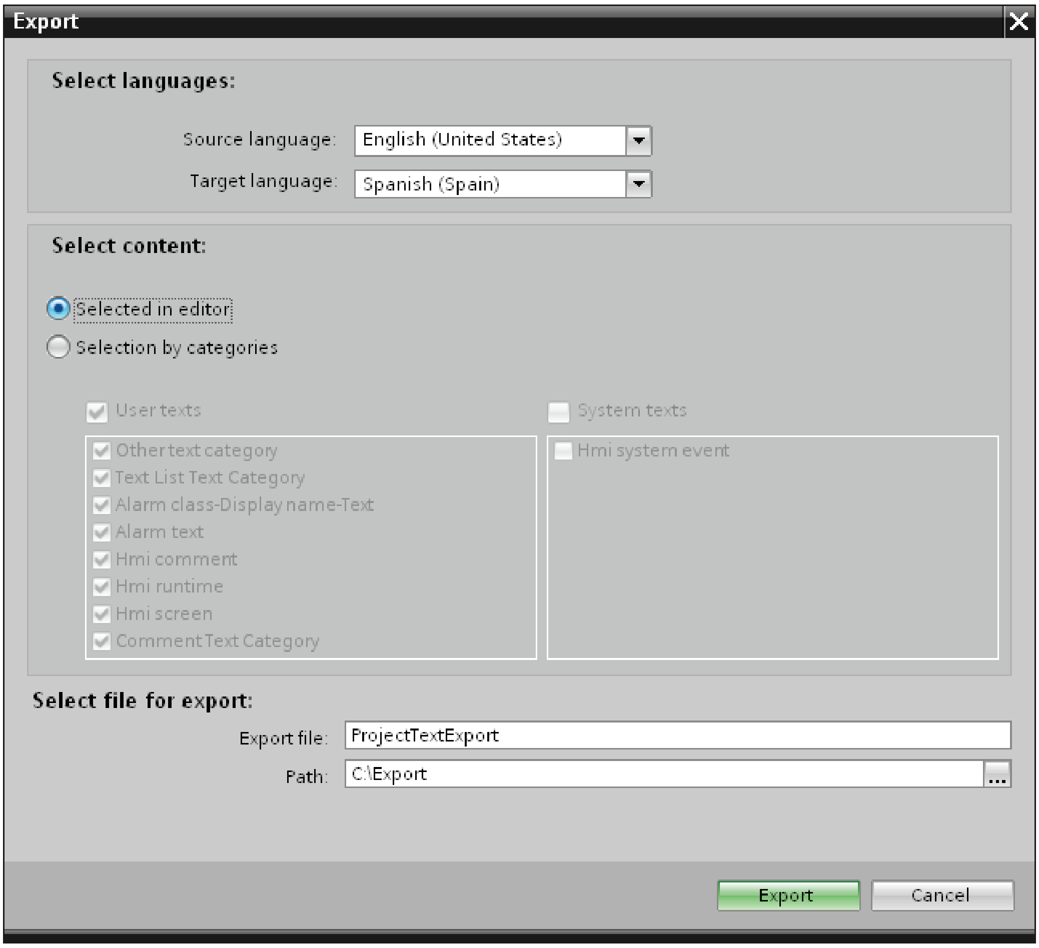
Exporting project texts

To export individual project texts, proceed as follows:

1. Click on the arrow to the left of "Languages & resources" in the project tree.

   The child elements are displayed.

2. Double-click on "Project texts". The "Project texts" editor will open.
3. Select the texts you want to export.
4. Click on the ➔ button. The "Export" dialog opens.



Export

Select languages:

Source language: English (United States)

Target language: Spanish (Spain)

Select content:

● Selected in editor

○ Selection by categories

☑ User texts

☑ Other text category
☑ Text List Text Category
☑ Alarm class-Display name-Text
☑ Alarm text
☑ Hmi comment
☑ Hmi runtime
☑ Hmi screen
☑ Comment Text Category

□ System texts

□ Hmi system event

Select file for export:

Export file: ProjectTextExport

Path: C:\Export

5. From the "Source language" drop-down list, select the language from which you wish to translate, for example Italian.
6. From the "Target language" drop-down list, select the language into which the texts are to be translated, for example, French.
7. Enter a file path and a file name for the export file in the "Export file" input field.
8. Click "Export".

Result

The texts selected in the "Project texts" editor are written to an xlsx file. The xlsx file will be stored in the specified folder.

You can alternatively select and export all project texts from categories. Select "User texts" or "System texts" in the "Export" dialog in line with the type of texts you wish to export. In this case, export can additionally be limited by categories.

Note

Project texts in library objects cannot be exported.

See also

Working with multiple languages (Page 2729)

Translating texts using reference texts (Page 2733)

Translating texts directly (Page 2731)

Project text basics (Page 2730)

Importing project texts (Page 2736)

Importing project texts

Edit the xlsx file or send it to a translator. Import the texts once they have been translated. The foreign languages will be imported to the relevant object in the project.

Requirements

- At least two languages have been enabled in the "Project languages" editor, for example, Italian and French.

Importing project texts

To import a project text file, proceed as follows:

1. Click on the arrow to the left of "Languages & resources" in the project tree.

   The lower-level elements will be displayed.
2. Double-click on "Project texts". The "Project texts" editor will open.
3. Click on the ← button. The "Import" dialog opens.
4. Select the path and file name of the import file from the "Import file" field.
5. Activate the "Import source language" check box if you have made changes to the source language in the export file and would like to overwrite the entries in the project with the changes.
6. Click on "Import".

Result

You have imported the project texts.

See also

Exporting project texts (Page 2734)

Working with multiple languages (Page 2729)

Project text basics (Page 2730)

Translating texts directly (Page 2731)

Translating texts using reference texts (Page 2733)

10.9.4.6 Using language-specific graphics

"Graphics" editor

Introduction

You use the "Graphics" editor to manage the configured graphic objects in different language versions. Multilingual projects sometimes also require language-specific versions of the graphics, for example, if

- the graphics contain text;
- cultural aspects play a role in the graphics.

Opening the "Graphics" editor

Double-click on "Languages and resources" in the project tree.

Work area

The work area displays all configured graphic objects in a table. There is a separate column in the table for each project language. Each column in the table contains the versions of the graphics for one particular language.

In addition, you can specify a default graphic for each graphic to be displayed whenever a language-specific graphic for a project language does not exist.

Preview

The preview shows you how the graphics will look on various devices.

See also

Storing external graphics in the graphics library (Basic, Advanced, Professional) (Page 2739)

Storing graphics in the graphics library (Basic, Advanced, Professional) (Page 2737)

Languages in WinCC (Page 2723)

Storing graphics in the graphics library (Basic, Advanced, Professional)

Introduction

You use the "Graphics" editor to import graphics you want to use in screens in the "Screens" editor. It also allows you to manage language-specific versions of graphics. A preview shows the graphic displays on various HMI devices.

Requirement

- The language-dependent versions of a graphic are available.
- Multiple languages have been enabled in the "Project languages" editor.
- The "Graphics" editor is open.

Inserting graphics

1. Click "Add" in the "Graphics library" table. A dialog box opens.
2. Select the required graphic file.
3. Click "Open" in the dialog box.

   The graphic will be imported to the project and displayed in all cells in this row in the "Graphics" editor.
4. Click in the corresponding cell of a language for which a language-dependent version of this graphic exists.
5. Select "Add graphic" from the shortcut menu. A dialog box opens.
6. Select the desired graphic file and click "Open."

   The language-dependent version is inserted in the table in place of the reference language graphic.
7. Then, in the "Default graphic" column, import a graphic to be displayed in runtime for those languages for which there is no language-specific graphic.

You can also drag&drop a graphic from Windows Explorer to the relevant position in the "Graphics library" table.

Displaying graphics in the HMI device preview

1. Click on a graphic in the table.
2. Select the required HMI device under "Properties > Graphics settings > Device preview" in the Inspector window.

   The graphic will then be displayed as it will appear in runtime on the selected HMI device.

Result

The graphics added are available in the "Graphics" editor. The graphic assigned to the respective editing language will be displayed during editing. The default screen will be displayed in all editing languages for which no screen has been imported.

The screens assigned to the respective runtime language are displayed during runtime. The default screen is displayed in all runtime languages for which a screen has not been imported.

Note

If you disable a project language, all of the graphic objects you have already created in this language will be deleted from the current project.

See also

"Graphics" editor (Page 2737)

Storing external graphics in the graphics library (Basic, Advanced, Professional)

Introduction

To display graphics that have been created in an external graphics program in your screens, you will first have to store these graphics in the graphics browser of the WinCC project.

Requirement

- Multiple languages have been enabled in the "Project languages" editor.
- The "Graphics" editor is open.
- There is a graphic in the "Graphics" editor.

Creating and adding a new graphic as an OLE object

1. Click "Add" in the "Graphics library" table. A dialog box opens.
2. Navigate to the folder in which the graphic is stored.
3. Click "Open" in the dialog box.

   The graphic will be imported to the project and displayed in all cells in this row in the "Graphics" editor.
4. Click in the corresponding cell of a language for which a language-dependent version of this graphic exists.
5. Select "Insert object" from the shortcut menu. The "Insert object" dialog box opens.

   Note

   In addition, the dialog "External application running..." will open. The dialog will not close until you exit the external application.

   ---
6. Select "Insert object > Create new" and an object type in the dialog.

7. Click "OK." The associated graphic program is opened.
8. Close the graphics program once you have created the graphic.

   The graphic will be stored in the graphic programming software standard format and added to the graphic browser.

Inserting created graphics in WinCC

1. Click in the corresponding cell of a language for which a language-dependent version of this graphic exists.
2. Select "Insert object" from the shortcut menu. The "Insert object" dialog box opens.

Note

In addition, the dialog "External application running..." will open. The dialog will not close until you exit the external application.

3. From the "Insert object" dialog box, select "Create from file."
4. Click the "Browse" button.
5. Navigate to the created graphic and select it.

Note

To import graphics files, note the following size restrictions:

*.bmp, *.tif, *.emf, *.wmf ≤4 MB

*.jpg, *.jpeg, *.ico, *.gif "*≤1 MB

Result

The OLE objects added are available in the "Graphics" editor.

Versions of the graphics for the current editing language are displayed in the "Screens" editor. The default graphic is displayed in all editing languages for which no screen has been imported.

The graphic is displayed in runtime in the set runtime language. The default graphic is displayed in all runtime languages for which no graphic has been imported.

You can double-click OLE objects in your library to open them for editing in the corresponding graphic editor.

See also

"Graphics" editor (Page 2737)

10.9.4.7 Languages in runtime

Languages in Runtime

Using multiple runtime languages

You can decide which project languages are to be used in runtime on a particular HMI device. The number of runtime languages that can be available at one time on the HMI device depends on the device. To enable the operator to switch between languages during runtime, you must configure a corresponding operator control.

When runtime starts, the project is displayed according to the most recent language setting. When runtime starts the first time, the language with the lowest number in the "Order for language setting" is displayed.

Setting runtime languages during configuration

In the "Languages and Fonts" editor you can specifiy:

- The project languages to be available as runtime languages for the respective HMI device.
- The order in which the languages are to be switched.

See also

Methods for language switching (Page 2741)

Enabling the runtime language (Page 2742)

Setting the runtime language order for language switching (Page 2744)

Setting the default font for a runtime language (Page 2746)

Selecting the log language (Page 2747)

Languages in WinCC (Page 2723)

Methods for language switching

Introduction

You need to configure language switching if you want to have multiple runtime languages available on the HMI device. This is necessary to enable the operator to switch between the various Runtime languages.

Methods for language switching

You can configure the following methods for language switching:

- Direct language selection

  Each language is set by means of a separate button. In this case, you create a button for each Runtime language.
- Language switching

  The operator switches the languages using a button.

Regardless of the method used, the button names must be translated into each of the languages used. You can also configure an output field that displays the current language setting.

See also

Languages in Runtime (Page 2741)

Selecting the log language (Page 2747)

Enabling the runtime language (Page 2742)

Setting the runtime language order for language switching (Page 2744)

Setting the default font for a runtime language (Page 2746)

Enabling the runtime language

Introduction

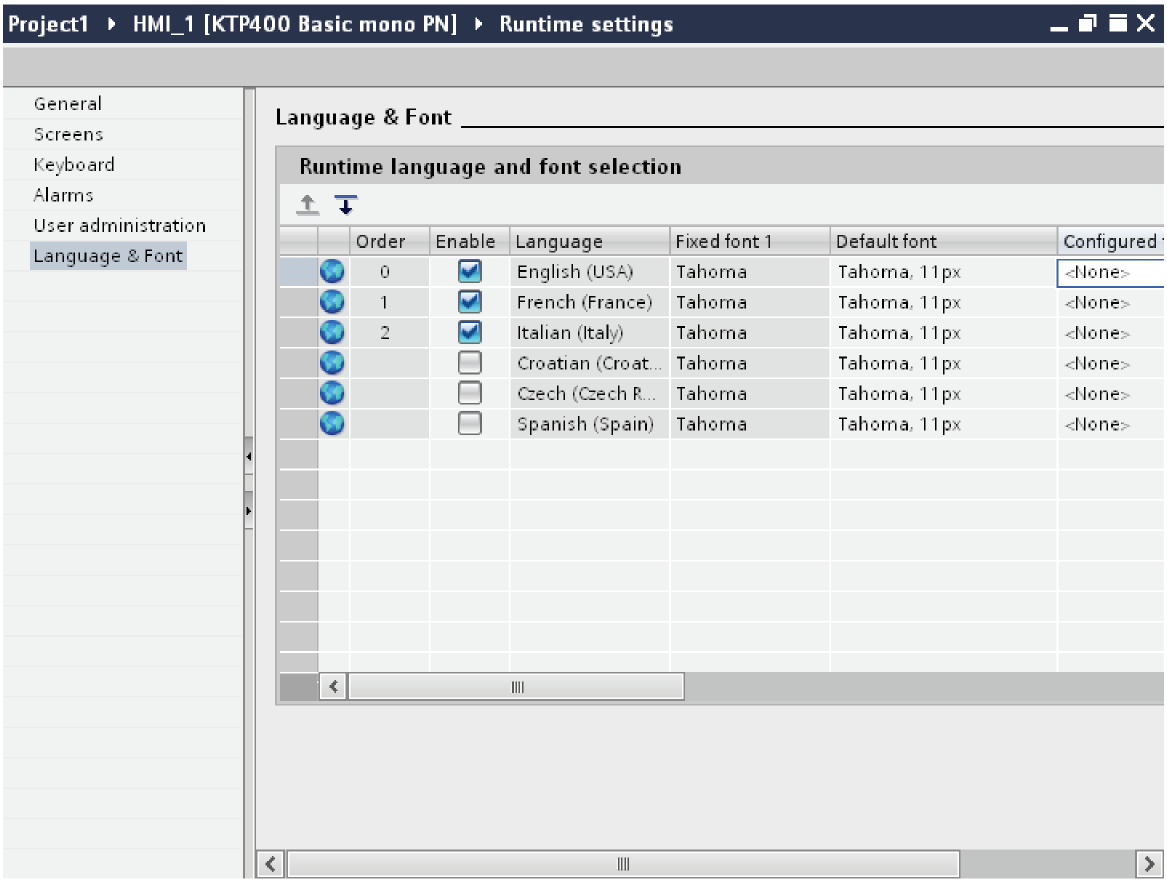
The "Language & Font" editor shows all project languages available in the project. Here you select which project languages are to be available as runtime languages on the HMI device.

Requirements

Multiple languages have been enabled in the "Project languages" editor.

Procedure

1. Double-click on "Runtime settings" in the project tree.
2. Click on "Language & Font".
3. Select the following languages:
   - German
   - Chinese
   - French



Language & Font

Runtime language and font selection

Order	Enable	Language	Fixed font 1	Default font	Configured
0	☑	English (USA)	Tahoma	Tahoma, 11px	<None>
1	☑	French (France)	Tahoma	Tahoma, 11px	<None>
2	☑	Italian (Italy)	Tahoma	Tahoma, 11px	<None>
	□	Croatian (Croat...	Tahoma	Tahoma, 11px	<None>
	□	Czech (Czech R...	Tahoma	Tahoma, 11px	<None>
	□	Spanish (Spain)	Tahoma	Tahoma, 11px	<None>

Result

You have now set three runtime languages. A number is automatically assigned to each language in the "Order" column. The enabled runtime languages are transferred with the compiled project to the HMI device.

If the number of languages selected exceeds the number that can be transferred to the HMI device, the table background changes color.

See also

Languages in Runtime (Page 2741)

Selecting the log language (Page 2747)

Setting the runtime language order for language switching (Page 2744)

Methods for language switching (Page 2741)

Setting the runtime language order for language switching

Introduction

You specify the language order for runtime language switching. The first time runtime starts, the project is displayed in the language with the lowest number in the "Order" column.

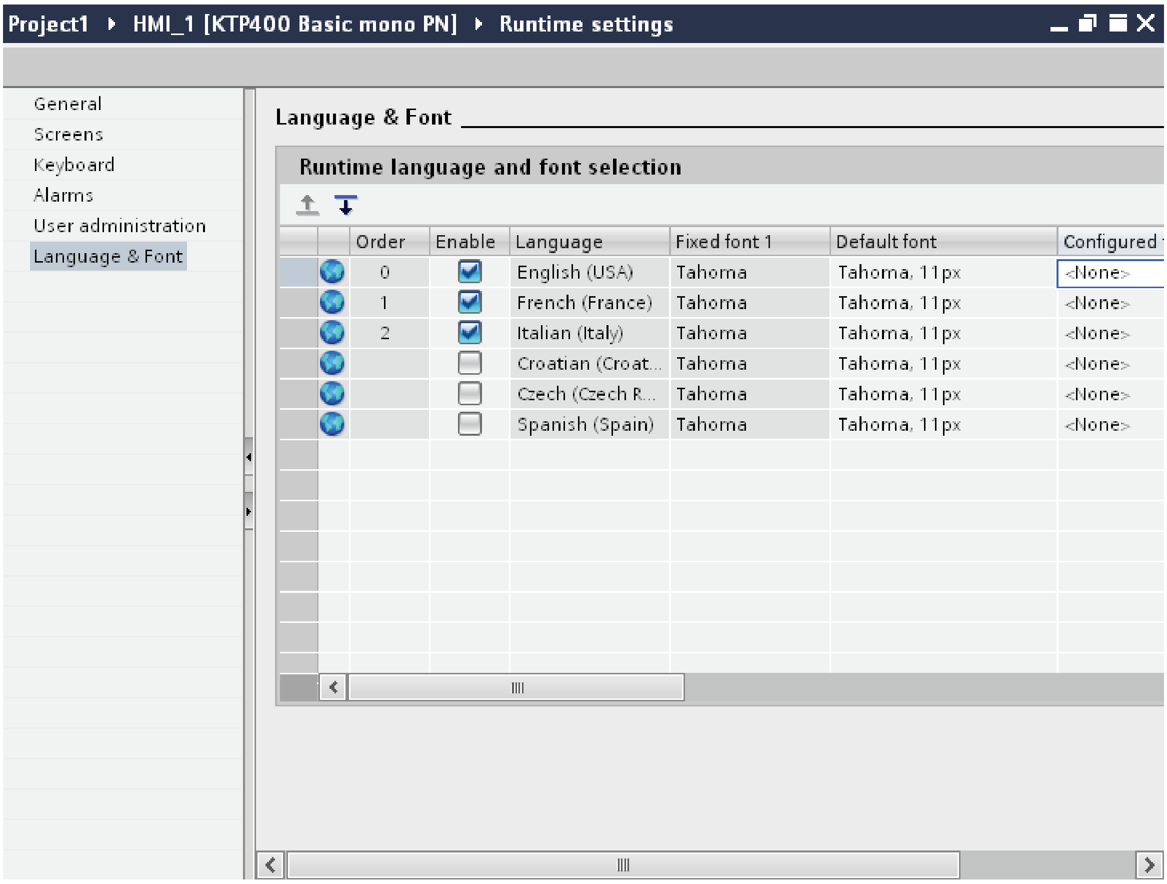
Requirements

- Multiple languages have been enabled in the "Project languages" editor.
- "Language & Font" is open in the editor and three runtime languages have been set in the following order:

  1. German
  2. Chinese
  3. French

Procedure

1. Select the runtime language "German".
2. Click the ⤓ button. The runtime language "German" will move down a place. The number will automatically be changed to "2" in the "Order" column.


Project1 ▸ HMI_1 [KTP400 Basic mono PN] ▸ Runtime settings
General
Screens
Keyboard
Alarms
User administration
Language & Font
Language & Font
Runtime language and font selection

Order	Enable	Language	Fixed font 1	Default font	Configured
0	☑	English (USA)	Tahoma	Tahoma, 11px	<None>
1	☑	French (France)	Tahoma	Tahoma, 11px	<None>
2	☑	Italian (Italy)	Tahoma	Tahoma, 11px	<None>
	□	Croatian (Croat...	Tahoma	Tahoma, 11px	<None>
	□	Czech (Czech R...	Tahoma	Tahoma, 11px	<None>
	□	Spanish (Spain)	Tahoma	Tahoma, 11px	<None>

Result

You have changed the order of runtime languages. The first time runtime starts, the project will be displayed in the language with the smallest number, in other words Chinese. If the language is switched, this will happen in numerical order.

See also

Languages in Runtime (Page 2741)

Selecting the log language (Page 2747)

Enabling the runtime language (Page 2742)

Setting the default font for a runtime language (Page 2746)

Methods for language switching (Page 2741)

Setting the default font for a runtime language

Introduction

You can specify the font used to display the texts for each runtime language on the HMI device in the "Language & Font" editor. The default font is used in all texts, such as dialog texts, for which you cannot define a specific font.

WinCC offers only fonts supported by the HMI device.

Requirements

- Multiple languages have been enabled in the "Project languages" editor.
- Three runtime languages have been enabled in the "Language & Font" editor.
  1. Chinese
  2. German
  3. French

Procedure

1. Double-click on "Runtime settings" in the project tree.
2. Click on "Language & Font". The table shows the runtime languages and fonts set.
3. Click in the "French" row in the "Default font" column.
4. Select the font to be used by default if a font cannot be selected for a given text.

Result

The project texts for the runtime language "French" are displayed on the HMI device in the selected font.

These fonts are transferred to the HMI device during a transfer operation.

The default font is also used for the representation of dialogs in the operating system of the HMI device. Select a smaller font as default if the full length of the dialog texts or headers is not displayed.

See also

Languages in Runtime (Page 2741)

Selecting the log language (Page 2747)

Setting the runtime language order for language switching (Page 2744)

Methods for language switching (Page 2741)

Selecting the log language

Introduction

In the "Runtime settings > General" editor, select the language to be used for writing to logs in runtime.

Requirements

- The languages used in your project are activated in the "Project languages" editor, for example "German" and "English" .

Procedure

1. Double-click on "Runtime settings" in the project tree.
2. Click on "Language & Font".
3. Activate the runtime languages, for example, "German" and "English".
4. Specify the "order":
   - 1 German
   - 2 English
5. Click on "Runtime settings > General".
6. Select "German" for "Logs > Log language".

Result

After loading, the project will start in the runtime language "German". The logs are now written in German. During runtime, the operator switches the runtime language to English. The logs will still to be written in German.

See also

Languages in Runtime (Page 2741)

Setting the default font for a runtime language (Page 2746)

Setting the runtime language order for language switching (Page 2744)

Methods for language switching (Page 2741)

Enabling the runtime language (Page 2742)

10.9.4.8 Example of multilingual configuration

Example: Configuring a button for language switching

Introduction

In this example, you configure a button that can be used to toggle between multiple runtime languages during runtime.

Requirements

- You have completed the "Configuring a button in multiple languages" example.
- The "Screen_1" screen is open.
- The button on the screen has been selected.

Procedure

1. In the Inspector window, select "Properties > Events > Press".
2. Click on "Add function" in the table.
3. Select the "SetLanguage" system function.

Result

You have assigned the button the function "SetLanguage". Pressing the button during runtime will switch the runtime language. The runtime languages are switched in the order specified by the number sequence in the "Languages and fonts" editor.

See also

Example: Configuring a button in multiple languages (Page 2748)

Example: Configuring a button for language switching for each runtime language (Page 2749)

Languages in WinCC (Page 2723)

Example: Configuring a button in multiple languages

Introduction

In this example, you configure a "Sprache umschalten" button in German and "Switch language" button in English.

Requirements

- The languages "German" and "English" have been enabled in the "Project languages" editor.
- German has been set as editing and reference language.
- You have created and opened the "Screen_1" screen.
- The Inspector window is open.

Procedure

1. Drag-and-drop a button from the "Tools" task card into the screen.

   The button will be added to the screen.
2. In the Inspector window, open "Properties > Properties > General".
3. Enter the text ""Sprache umschalten" under "Text > off".
4. Press <Enter> to confirm. The button is named.
5. Open the "Tasks" task card.
6. Select "English" under "Languages & Resources > Editing language".
7. Enter the label "Switch Language" under "Properties > Properties > General > Text > Off" in the Inspector window.

Result

The button name is configured in German and English language. The button name corresponding with the current Runtime language is shown in Runtime.

See also

Example: Configuring a button for language switching (Page 2748)

Example: Configuring a button for language switching for each runtime language (Page 2749)

Example: Configuring a button for language switching for each runtime language

Introduction

In this example, you configure a "Sprache umschalten" button in German and "Switch language" button in English.

Requirements

- The following languages have been enabled in the "Project languages" editor
  - German
  - English
  - Italian
- All languages have been set as runtime languages in the "Runtime settings > Language & Font" editor.
- You have created and opened the "Screen_1" screen.
- Three buttons have been created on the screen:
  - Button_1 labelled "Deutsch"
  - Button_2 labelled "English"
  - Button_3 labelled "Italiano"
- The Inspector window is open.

Procedure

1. Select "Button_1".
2. In the Inspector window, select "Properties > Events > Press".
3. Click on <Add function> in the table.
4. Select the "SetLanguage" system function.
5. Click on the "Switch" field.
6. Click on the button.
7. Select "Runtime language". The field will be highlighted in red.
8. Select "German" from the drop-down list.
9. Repeat steps 1 - 8 for the other two buttons and select the corresponding runtime language.

Result

You have configured three buttons for switching language in runtime. Each button will switch to a different runtime language. For example, clicking on the "English" button during runtime will switch the runtime language to English.

See also

Example: Configuring a button for language switching (Page 2748)

Example: Configuring a button in multiple languages (Page 2748)

10.9.5 Replacing devices

10.9.5.1 Basics

Introduction

You can use existing configurations for new devices and optimize these configurations with little manual effort.

All data configured by you is retained in the configuration data. This means you do not need to copy individual objects of one device and paste them to another.

Principle

The following applies when you replace devices:

- Only functions supported by the new device are available. Only configuration data supported by the new device are displayed.

  This affects

  – recipes,

  – objects available on the screens,

  – available system functions,

  – available communication logs, etc.

- The number of supported objects, such as screens or tags, may be limited on the new device. If the existing objects exceed the limitations on the new device, the objects are displayed in full. The objects are, however, highlighted in color in the individual editors. An error is generated when the project data is compiled.

  Manual post processing is required when switching to a device with fewer features.

  Example: Limited number of connections

  All connections will be highlighted in color as invalid if fewer connections are supported on the new device than have been configured. Delete any excess connections.

Note

If you replace a Panel and select a PC Station as your new device, for example, WinCC Runtime Advanced will automatically be moved below the PC Station in the project tree.

See also

Device-specific functions (Page 2752)

Screen adjustment options (Page 2755)

Key assignment when replacing devices (Page 2753)

Engineering system (Page 2848)

Basic Panel (Page 2850)

10.9.5.2 Device-specific functions

Device-specific functions

Functions dependent on the device

Functions dependent on the device are implemented as follows:

- Colors

  The color is changed automatically when you switch from a device with full color display to one with a smaller color range.

  If you change the color manually and then change back again to a device with a larger range of colors, the reduced range of colors will be retained.
- Fonts

  Any configured font not available on a device will be replaced by a similar one or by the configured default font. The default font depends on the device selected.
- Character sets with different font sizes

  Avoid using too many different font sizes when configuring the following devices:

  – OP 73

  – OP 77A

  – TP 177A

  A character set is downloaded to the device for each font size. Check the Inspector window during compilation to see how much of the device memory is being used by the character sets.

- Font size

  Use small Windows fonts to display the text on devices. If you use large Windows fonts, then, depending on the size of the display, the text will not be displayed in full.

  Using font sizes of 28 pixels or more for the OP 77A and TP 177A devices will affect device performance.

  The character scope is much greater for Asian languages. The use of different font sizes therefore has serious implications on the memory requirements of all devices.

  Use the same font type for all large characters throughout the project to ensure effective and efficient configuration.

- Screens and screen objects

  If the new device supports a different resolution than the previous device when you replace a device, there are several ways to adjust the screens.

  Adjust the size of the screens to the new device in the menu under "Options > Settings > Visualization > Fit to size screen".

See also

Basics (Page 2751)

Key assignment when replacing devices

Introduction

The devices available each have different function keys. The functions configured for these keys will be mapped to the available function keys of the new device if the device is replaced.

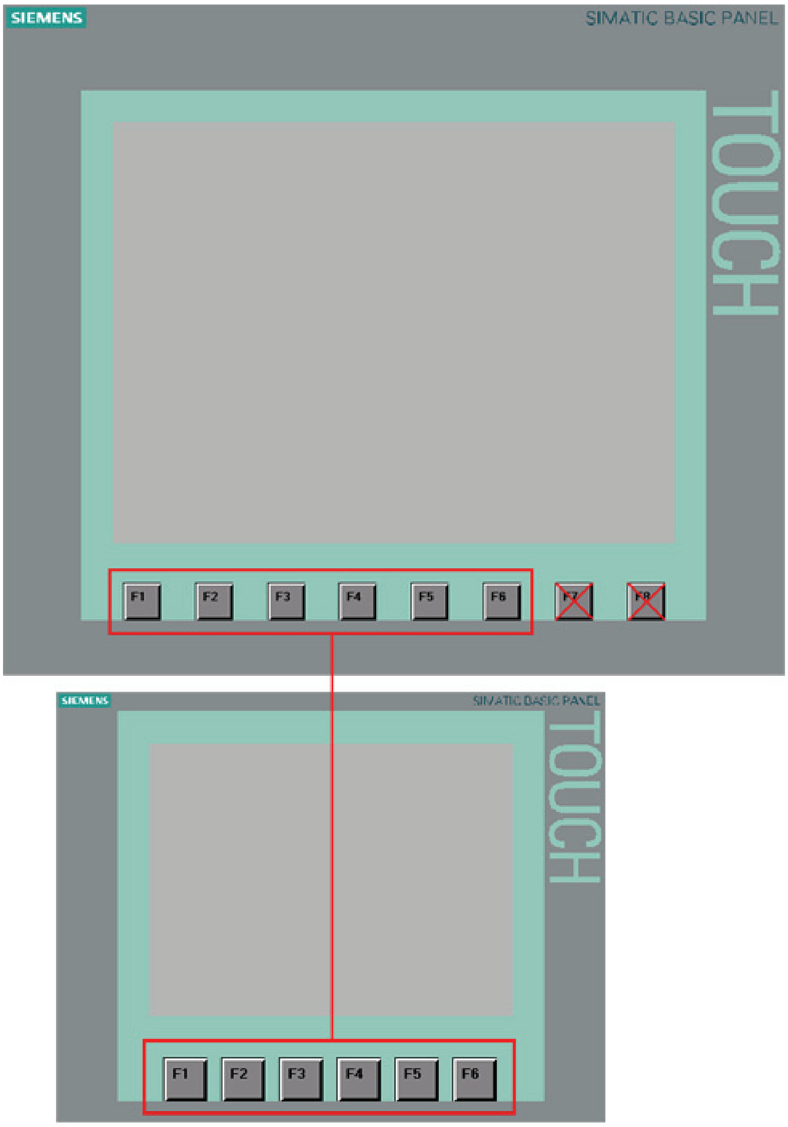
Function key mapping

The function keys below the display are mapped from left to right to the new device. If the new device has fewer keys, the keys it does not have are not mapped.

Example: Replacing a KTP1000 Basic with a KTP600 Basic

You have configured a function for F2 in KTP1000 Basic. This function is triggered by F2 following replacement with a KTP600 Basic.

If you have used F7 in a KTP1000 Basic, this function will no longer be available if the panel is replaced with a KTP600 Basic.



Mapping of control keys and cursor keys

The following keys are mapped only to the same keys of the new device:

- HELP
- ESC
- ACK
- ENTER
- PAGE UP
- PAGE DOWN
- CURSOR UP
- CURSOR DOWN

See also

Basics (Page 2751)

Screen adjustment options (Page 2755)

10.9.5.3 Adjusting screens to the new device

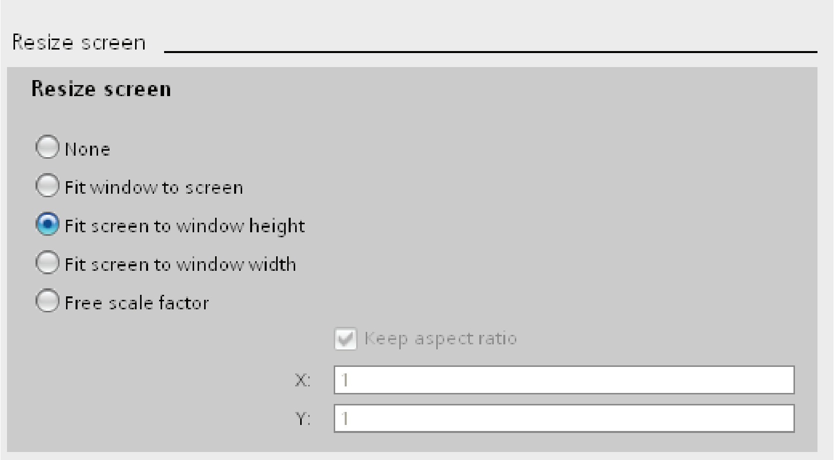
Screen adjustment options

Introduction

Select fit to size for screens before you replace a device. Fit to size is particularly important when switching devices with different display resolutions.

Screen adjustment when replacing devices

Adjust the size of the screens to the new device in the menu under "Options > Settings > Visualization > Fit to size screen".



Resize screen

Resize screen

○ None
○ Fit window to screen
● Fit screen to window height
○ Fit screen to window width
○ Free scale factor

☑ Keep aspect ratio

X: 1

Y: 1

Select one of the following settings.

None

The screens are not scaled. The objects in the screen retain their position and size.

This option may result in objects being outside the configurable area if the display of the new device is smaller than the old one.

Adjusting the width and height to the new device

The position and object size are adjusted to the new display size. Adjustment takes place along the x-axis and the y-axis. Graphics and font size are adjusted accordingly.

Object adjustment to content can be prevented for objects such as graphic views or text fields.

Note

The objects are distorted if you replace a device with a landscape format display with a device with a portrait format display. The difference in display format can, for example, result in object labels being cut off and content not being fitted to the object. You must therefore adjust the screens to the new device once you have replaced devices.

Fit screen to window height

The aspect ratio is maintained and the screens are adjusted to the height of the new device.

Use this option when you are replacing a device with display format 4:3, for example, with a device with widescreen.

Fit screen to window width

The aspect ratio is maintained and the screens are adjusted to the width of the new device.

Use this option when you are replacing a device with widescreen, for example, with a device with display format 4:3.

Free scale factor

You select a free scale factor for screen adjustment. You can specify a factor for the x-axis and the y-axis.

Using a free scale factor of < 1 may distort the objects. Object labels may, for example, be cut off and the content may not be fitted to the object.

You must therefore adjust the screens to the new device once you have replaced devices.

Note

The aspect ratio is not adjusted for objects with a fixed aspect ratio, for example, gauge, circle. The objects are displayed on the new device with the same aspect ratio as prior to the replacement of the device.

See also

Specifying the position of screen objects (Page 2758)

Fit objects to contents (Page 2757)

Basics (Page 2751)

Key assignment when replacing devices (Page 2753)

Fit objects to contents

Introduction

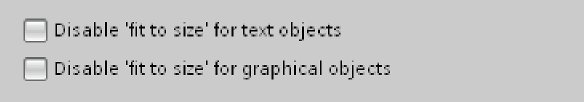
For some objects, you can specify fit to respective content in the Inspector window, for example:

- Text field: fit to text content
- I/O field: fit to text content
- Symbolic I/O field: fit to text content or to text list
- Graphic view: fit to included graphic

Fit to size for text and graphic objects

Disable automatic fit to size of the individual objects in the menu under "Options > Settings > Visualization > Resize screen and screen objects > Fit to content". This results in scaling of the objects as specified under "Options > Settings > Visualization > Resize screen and screen objects".

Select the objects which are not automatically fitted to size.



□ Disable 'fit to size' for text objects

□ Disable 'fit to size' for graphical objects

- If "Disable 'fit to size' for text objects" is activated, automatic fit to size is ignored in the text object properties.

  If you have activated "Fit screen to window height", the text field along with the other objects is scaled in accordance with the height of the new device.
- If "Disable 'fit to size' for graphic objects" is activated, automatic fit to size is ignored in the graphic object properties.

  If you have activated "Fit screen to window width", the graphics view along with the other objects is scaled in accordance with the width of the new device.

Note

The settings have no effect on screen objects whose size cannot be changed, such as alarm indicators or screen objects with a fixed aspect ratio.

"Disable 'fit to size' for text objects" and "Disable 'fit to size' for graphic objects" have no effect if:

- You have activated "Resize screen and screen objects > None".

- You have activated "Fit screen to window width and height" and the new device has the same resolution as the current device.

- You have activated "Fit screen to window height" and the new device has the same resolution as the current device.

- You have activated "Fit screen to window width" and the new device has the same resolution as the current device.

See also

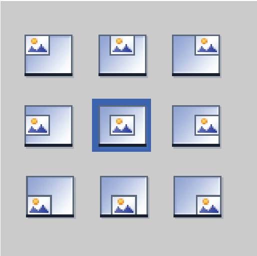
Specifying the position of screen objects (Page 2758)

Screen adjustment options (Page 2755)

Specifying the position of screen objects

Introduction

There are various ways to adjust the position of screen objects to the new device.



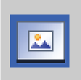
Select position

Adjust the position of the screen objects to the new device in the menu under "Options > Settings > Visualization > Fit to size screen > Select position".

Example

The following option aligns the objects with the top left edge.

The following object centers the objects in the middle of the screen.



See also

Fit objects to contents (Page 2757)

Screen adjustment options (Page 2755)

10.9.6 Copying between devices and editors

10.9.6.1 Basics

Basics

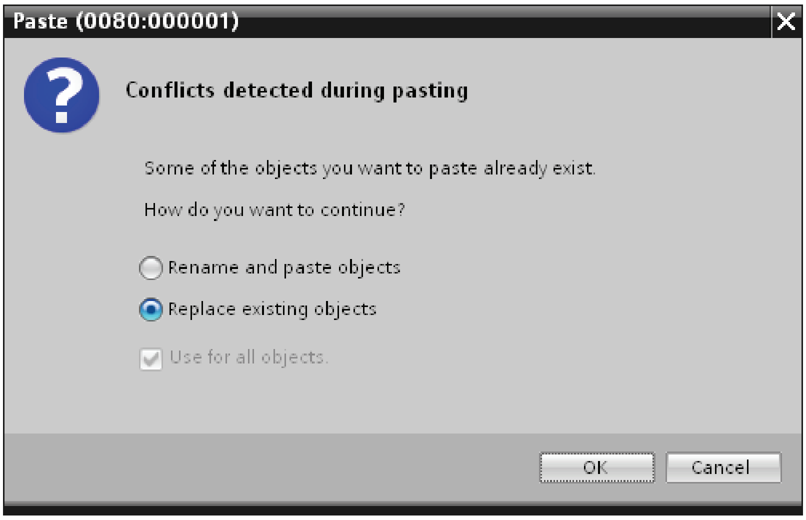
Copying and pasting within an HMI device

You can copy and paste objects, such as display objects, within an HMI device. If the object is already created in the editor, when the object name is inserted a number is automatically attached, in accordance with the following principle:

- "<Object_name>_1" is renamed to "<Object_name>_2".
- "<Object_name>_2" is renamed to "<Object_name>_3".

Copying and pasting between HMI devices

You can also copy and paste between HMI devices. If an object with the same name already exists, you have the following options:



Paste (0080:000001)

Conflicts detected during pasting

Some of the objects you want to paste already exist.

How do you want to continue?

○ Rename and paste objects

● Replace existing objects

☑ Use for all objects.

Note

Exception to this basic rule

Copying and pasting of an alarm class that has been generated from project-wide alarm class is handled differently than with this basic rule. When the copied alarm class already exists in the target HMI device within the same project, the "Paste" command is not performed.

Copying user-defined folders

You can create user-defined folders for editors, for example, for HMI tags, screens, etc.

You can copy user-defined folders and paste them into another HMI device. The objects contained in a user-defined folder may exceed the limitations applying to the other HMI device, such as the number of supported screens. After they have been pasted, all the objects are displayed. An error is displayed when the project data is compiled.

System folders cannot be copied.

See also

Unsupported objects and functionalities (Page 2760)

Unsupported objects and functionalities

Introduction

When an object is copied, all its properties and settings are transferred to the target HMI device.

Unsupported objects

Objects that are not supported in the target HMI device cannot be pasted.

Note

When you copy a screen containing objects which are not supported by the destination HMI device, the objects remain in the background. When you copy the screen again and the new device supports the objects, they are displayed again.

Invalid objects

The following objects become invalid once they have been pasted into the target HMI device:

- Referenced objects that do not exist in the target HMI device.
- Objects with settings that are not supported in the target HMI device.
- System functions that were configured for objects and that are not supported in the target HMI device.

Invalid objects are highlighted by a color coding. Select a supported object or create a new one. If you retain an invalid object, an error will be displayed when the project data is compiled.

Colors and fonts

Colors and fonts are supported to varying degrees by HMI devices. When unsupported colors and fonts are pasted, they are replaced by supported colors and fonts. When you paste the same object back into the source HMI device, the original settings become active again.

See also

Basics (Page 2759)

10.9.6.2 Copying and pasting

Copying screens

Introduction

You copy one or more screens from the "Screens" folder and paste them into the "Screens" folder of another device.

Type and size of the displays

In the case of HMI devices with keys, the available keys are displayed automatically in the screen. When a screen is copied between HMI devices, the keys are either displayed or hidden; functions configured for function keys are not transferred.

If there is less space for the screen in the target HMI device than in the source HMI device, you can adjust the size of and the spacing between existing objects.

Automatic fit to size for objects

1. Select "Options > Settings > Visualization > Resize screen and screen objects" in the menu.
2. Activate, for example, "Fit screen to window width and height".

See also

Copying recipes within an HMI device (Page 2762)

Copying objects with linked objects (Page 2763)

Linked objects copied automatically (Page 2764)

Copying recipes within an HMI device

"Recipes" Editor

You can copy recipes, recipe elements and recipe data records within each table. You copy a recipe element to another recipe.

Only WinCC Runtime Professional: You can copy a recipe view element to another recipe view. If a recipe view element of the same name already exists, a conflict dialog is displayed. You can select whether to replace or rename the recipe element. You can copy recipe elements to the first empty row of the "Recipe views" editor, "Elements" tab.

You can copy a recipe data record to another recipe, if the other recipe contains the same number of recipe elements. If the data types differ, the value will be copied to the target data record but it is assigned an error flag.

"Tags" editor

You can drag-and-drop a tag to a recipe element in the "Tag" column. The tag is linked to the recipe element. If a tag is already linked, an error message will be generated.

"Screens" editor

If you drag-and-drop a recipe to a screen, a new recipe display will be created and linked to the recipe.

See also

Copying screens (Page 2761)

Copying objects with linked objects (Page 2763)

Linked objects copied automatically (Page 2764)

Copying objects with linked objects

Introduction

An object is linked to another object in the following situations, for example:

- You specify a tag for an alarm as a trigger tag.

  The alarm is the object. The tag is the linked object.
- You specify a connection for an external tag.

  The tag is the object. The connection is the linked object.

The object is always fully inserted during copying and pasting. Whether or not the linked object is pasted depends on the command used to insert it.

Simple pasting

The linked object is not copied. The linked object is transferred and handled as follows in the target HMI device:

- If an object with the same name exists, the existing object with its settings is used.
- If no object with the same name exists, the name of the object will be displayed. The object becomes invalid.

For some objects, linked objects are pasted automatically during simple pasting.

Extended pasting

Select the "Extended paste" command in the shortcut menu to paste the linked objects as well. If objects of the same name exist in the target HMI device, you need to decide whether or not to overwrite each of these objects.

See also

Copying screens (Page 2761)

Copying recipes within an HMI device (Page 2762)

Linked objects copied automatically

Copying linked objects

The following table shows the objects for which linked objects are pasted automatically in simple pasting.

Object	Linked object
Screen	Template
Symbolic I/O field	Text list
Graphic I/O field	Graphics list
Graphic view	Graphic
Tag	Alarm
	Cycle
Recipe element	Text list
Scheduler	Triggers

See also

Copying screens (Page 2761)

Copying recipes within an HMI device (Page 2762)

Drag & drop from the details view

Introduction

You can improve configuration efficiency with just a few simple measures. Below are a few examples of efficient configuration.

Pasting objects to a screen from the details view

You can drag objects in the details view from various different editors to other editors.

Pasting a symbolic I/O field

1. Open a screen.
2. Click on the "Text and graphics lists" editor in the project tree. All existing text and graphics lists will be shown in the details view.
3. Click on a text list, for example, "Textlist1" in the Details view.
4. Drag-and-drop a text list from the Details view to a screen. A symbolic I/O field has been created and connected to the text list "Textlist1".

Pasting a graphic I/O field

1. Open a screen.
2. Click on the "Text and graphics lists" editor in the project tree. All existing text and graphics lists will be shown in the details view.
3. Click on a graphics list in the Details view, for example "Graficlist1".
4. Drag-and-drop a graphics list from the Details view to a screen. A graphic I/O field has been created and connected to the graphics list "Graficlist1".

Pasting an I/O field

1. Open a screen.
2. Click on the "HMI tags" editor in the project tree. All existing HMI tags will be shown in the Details view.
3. Click on an HMI tag in the Details view, for example "Tag1".
4. Drag-and-drop the HMI tag from the Details view to a screen. An I/O field has been created and connected to the HMI tag "Tag1".

10.10 Compiling and loading

10.10.1 Compiling and loading projects

10.10.1.1 Overview of compiling and loading projects

Overview

The project is compiled in the background even as you are configuring it in WinCC. This reduces the time for final compilation. When you start compilation, you create a file that can be run on the corresponding HMI device.

If an error occurs during compilation, WinCC provides support in locating and correcting it.

Once you have corrected any problems, you download the compiled project to the HMI devices on which the project is to run.

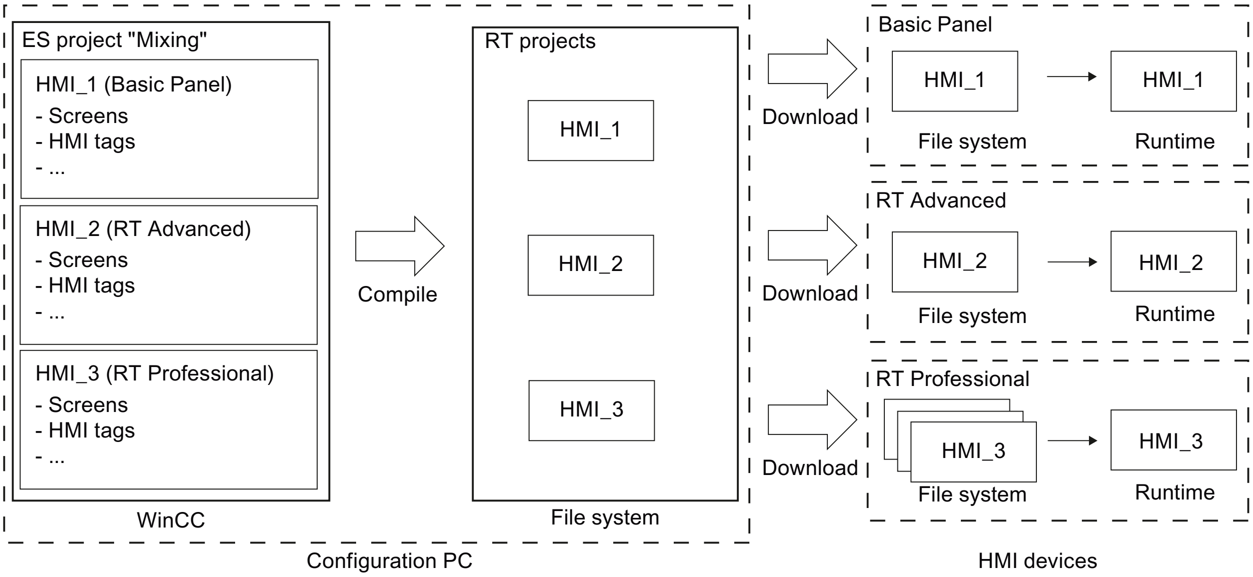
If you are using HMI tags in your project that are connected to PLC tags, you should also compile all modified S7 blocks with the command "Compile > Software" in the shortcut menu before you compile the HMI device.

Definition of terms

The term "project" has two different meanings in the contexts of compilation and loading. "Project" is the WinCC project on the configuration PC. "Project" is also the Runtime project you create by compiling the configuration data of an HMI device and download to the HMI device.

- WinCC project: contains the configuration data of one or more HMI devices
- Runtime project: contains the compiled configuration data of an HMI device

The figure below illustrates the link between WinCC projects and Runtime projects using the example of the "Compile and load" process:



10.10.1.2 Compiling a project

Introduction

The changes made to the project are compiled in the background even as you are configuring a project in WinCC. Projects are compiled automatically when you load them. This ensures that the latest version of the project is loaded at all times.

WinCC checks consistency of the project during compilation. The error locations in the project are listed in the Inspector window. You can jump directly to the source of the error from the entry in the Inspector window. Check and correct errors found.

Scope of the compilation

Configuration data is compiled in the background as soon as you start configuring an HMI device. If you compile a project manually, only the changes in the configuration made since the last compilation process are compiled in the background.

You can start complete project compilation manually at any time; this may, for example, be done to test the consistency of the configured data.

Requirement

- A project is open.

Procedure

Proceed as follows to compile a project:

1. If you want to compile several HMI devices at the same time, select all the relevant HMI devices with multiple selection in the project tree.
2. Compile the project:
   - For delta compilation of the project, select the "Compile > Software" command from the shortcut menu of the HMI device.
   - To compile all project data, select the "Compile > Software (compile all)" command from the shortcut menu.

Result

The configuration data of all selected HMI devices is compiled. Any errors that occur during compilation are shown in the Inspector window.

10.10.1.3 Loading projects

Overview for loading of projects

Overview

Delta data of the project is automatically compiled before you download it to one or several HMI devices. This always ensures that the latest version of the project is transferred.

Loading a project to an HMI device

The following steps are completed prior to downloading:

1. The download settings are verified. The "Extended loading" dialog box is opened automatically during the initial download of a project to an HMI device. You use this dialog to define the protocol and interface or destination path for the project in accordance with the HMI device Runtime used.

   You can call the "Extended loading" dialog at any time with the menu command "Online > Advanced download to device...".

   The "Load preview" dialog opens.
2. The project is compiled. Warnings and errors during compilation are displayed in the Inspector window and in the "Load preview" dialog.

3. The "Load preview" dialog shows you the following information for each HMI device:
   - The individual steps for loading
   - Presettings that take effect at loading. You can change the default settings for this download process, if necessary.
   - Warning events (optional). You can download a project while ignoring the "warnings". The functionality may be restricted in Runtime.
   - Error events (optional). You cannot load the project. Eliminate the errors and then reload the project.

     WinCC will open the invalid configuration in the corresponding editor if you double-click the error message in the Inspector window. Correct the errors and reload the project.

If you are using HMI tags in your project that are connected to PLC tags, you should also compile all modified S7 blocks with the command "Compile > Software" in the shortcut menu before you compile the HMI device.

Loading with S7 routing

Configure the S7 routing settings in the "Devices & Networks" editor in the relevant PLC. The settings depend on the device configured.

S7 routing supports the following protocols:

- MPI/PROFIBUS
- Ethernet

Loading a project

Introduction

Before a project can run on an HMI device, you must first load it to the HMI device. During loading, you must most importantly specify whether existing data on the HMI device such as "user administration" and "recipe data" is to be overwritten.

If the HMI device supports PROFINET, the name of the HMI device entered in the project tree is used as the device name for PROFINET communication. The name is written to the HMI device during loading. If a device name for PROFINET communication has already been entered in the HMI device, it will be overwritten.

As a general rule, only one project can be active in Runtime on an HMI device. An HMI device is generally configured to exit Runtime automatically when loading is started. If this is not the case, you will have to exit Runtime manually on the HMI device.

Please refer to the documentation for the HMI device used for more detailed information on transfer settings.

Requirement

- You have created an HMI device in the project.
- The HMI device is connected to the configuration PC.
- Transfer mode is set in the HMI device.

Procedure

Proceed as follows to load a project:

1. To download a project simultaneously to several HMI devices, select the HMI devices by means of multiple selection in the project navigation.
2. Select the "Download to device > Software" command from the shortcut menu of the HMI device.
3. If the "Extended loading" dialog is open, configure the "Settings for loading". Make sure that the "Settings for loading" correspond to the "Transfer settings in the HMI device".
   - Select the protocol used, for example, Ethernet or HTTP.
   - Configure the relevant interface parameters on the configuration PC.
   - Make any interface-specific or protocol-specific settings required in the HMI device.
   - Click "Download".

   You can call the "Extended loading" dialog at any time with the menu command "Online > Advanced download to device...".

   The "Load preview" dialog opens. The project is compiled at the same time. The result is displayed in the "Load preview" dialog.
4. Check the displayed presettings and change them as necessary.
5. Click "Download".

Result

The project is loaded to all selected HMI devices. Any existing project is replaced. The data for user administration and / or recipes is replaced in accordance with the settings in the "Load preview" dialog. If errors or warnings occur during the download, corresponding alarms are displayed under "Info > Load" in the Inspector window.

On completion of the successful download of the project, you can execute it on the HMI device.

See also

Backing up and restoring data of the HMI device (Page 2779)

Updating the HMI device operating system (Page 2781)

Error messages during loading of projects (Page 2782)

Adapting the project for another HMI device (Page 2784)

Establishing a connection to the HMI device (Page 2785)

10.10.1.4 Runtime start

Starting Runtime on the HMI device

Introduction

On completion of the project download to the HMI device, you can start the project in Runtime. The project is saved in the HMI device to a file with the following extension:

- Basic Panels as well as OP 73, OP 77A and TP 177A: "*.srt"
- All other HMI devices: "*.fwc"

The project settings defined in the "Runtime settings" of the HMI device are activated when the project is started in Runtime.

The programs that you can use to start projects on the HMI device are available in the Runtime installation folder.

Requirements

WinCC Runtime is installed on the HMI device.

Procedure

The "RT Loader" application is started on a panel. The project loaded is started automatically after expiration of the configured delay.

If the project does not start automatically:

1. To select the project file, click "Settings" and enter the path to the project file under "Configuration file".
2. Click "OK" and then "Start".

10.10.2 Simulating projects

10.10.2.1 Simulation basics

Introduction

You can use the simulator to test the performance of your configuration on the configuration PC. This allows you to quickly locate any logical configuration errors before productive operation.

You can start the simulator as follows:

- In the shortcut menu of the HMI device, or in a screen: "Start simulation"
- Menu command "Online > Simulation > [Start|With tag simulator|With script debugger]"
- Under "Visualization > Simulate device" in the portal view.

Requirements

The simulation/runtime component is installed on the configuration PC.

Field of application

Use the simulator to test various functions of the operator control and monitoring system, such as:

- Checking limit levels and alarm outputs
- Consistency of interrupts
- Configured interrupt simulation
- Configured warnings
- Configured error messages
- Check of status displays
- Interconnection and screen layout

See also

Simulating a project (Page 2773)

10.10.2.2 Simulating a project

Introduction

You simulate your project with one of the following two methods:

- Without a connected PLC

  You change the value of area pointers and tags in a tag simulator that is read for the simulation of WinCC Runtime.

- With a connected PLC without a running process

  You simulate your project by running it directly in Runtime. The tags and area pointers become active. This allows you to create an authentic simulation of your configured HMI device in Runtime.

Note

Simulation restrictions

You cannot simulate the following system functions:

- CalibrateTouchScreen

You cannot simulate the Media Player. A static screen appears in the simulation window instead of the Media Player.

File access via scripts is not possible for HMI devices with Windows CE.

Requirement

- Simulation without a connected PLC: Tags have been created
- Simulation with a connected PLC but no active process: A project with tags and area pointers has been created

Procedure

To simulate a project using the tag simulator, follow these steps:

1. Open the project on the configuration PC.
2. Select the "Online > Simulation > With tag simulator" menu command.

   For initial project simulation, the simulator is started with a new, empty table. The project is opened simultaneously in Runtime.

   Toggle between the tag simulator and Runtime using the <Alt +Tab> key combination.
3. To simulate a process value, select the corresponding "tag" from the tag simulator.

   The table lists all configured tags. You can simulate up to 300 tags simultaneously.
4. Select the simulation mode in the "Simulation" column.
5. Change the value of tags and area pointers in the respective columns.

6. Activate the "Start" check box to start the simulation for this tag.
7. To save the simulation, select the menu command "File > Save" and enter a descriptive name, for example, "Mixing".

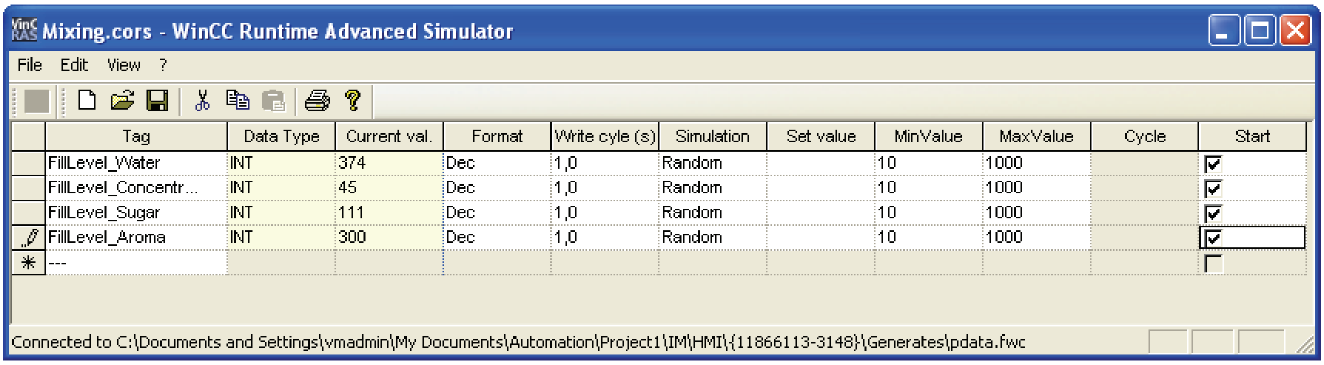
   The file name is assigned the extension "*.cors".

Result

The process values are simulated in Runtime. The tag values are created at random, or incremented, depending on the simulation mode.

To specify tag values, change the simulation mode to "<Display" and enter a value at "Set value".

The following figure shows a tag simulator with four tags whose values can be determined at random in a range of values from 10 to 1000:



Tag	Data Type	Current val.	Format	Write cyle (s)	Simulation	Set value	MinValue	MaxValue	Cycle	Start
FillLevel_Water	INT	374	Dec	1,0	Random		10	1000		☑
FillLevel_Concentr...	INT	45	Dec	1,0	Random		10	1000		☑
FillLevel_Sugar	INT	111	Dec	1,0	Random		10	1000		☑
FillLevel_Aroma	INT	300	Dec	1,0	Random		10	1000		☑
---										☐

Connected to C:\Documents and Settings\vmadmin\My Documents\Automation\Project1\IM\HMI\{11866113-3148}\Generates\pdata.fwc

Managing simulation data

If you have saved data from a previous simulation, you can open the file at a later point in time and simulate your project again. The tags and area pointers listed in the tag simulator must still be available in the project.

Proceed as follows to open a simulation file:

1. Select the menu command "Online > Simulate Runtime > With tag simulator".
2. Select the menu command "File > Open" in the tag simulator.
3. Select the corresponding simulation file and click "Open".

   The simulator loads the stored data.

Enabling and disabling tags

Start and stop the simulation for each tag separately in order to facilitate the transition from offline to online engineering. Activate "Start" in the corresponding row.

If a tag is activated, the simulation values are calculated and transferred to the WinCC simulator.

Deleting a tag

To delete a tag from the tag simulator, follow these steps:

1. Select the cell that contains the tag name.
2. Select the "Edit > Cut" menu command.

   The tag is removed from the table.

See also

Simulation basics (Page 2772)

Working with the tag simulator (Page 2775)

10.10.2.3 Working with the tag simulator

About the tag simulator

The tag simulator has the following columns:

Column	Description
Tag	Specifies the tags for the simulation.
Data type	Shows the data type of the selected tag.
Current value	Shows the simulated value of the defined tags.
Format	Specifies the selected format in which the tag values are simulated: • Decimal (1, 2, 3, 4, ...) • Hexadecimal (03CE, 01F3, ...) • Binary (0 and 1)
Write cycle	Specifies the selected time interval at which the current tag values are simulated. If you enter "2", for example, the current value of the tag will be shown every 2 seconds.
Simulation	Shows the method by which the tag values are processed during simulation.
Set value	Sets the selected value for the respective tag. The simulation start with the specified value.
minValue maxValue	Specifies the value range of the tag. You set a minimum and maximum value for this range. The default values are -32768 for the minimum and 32767 for the maximum.
Period	Contains the period during which the value of the tag is repeated for the "Increment" and "Decrement" simulation modes.
Start	Starts simulation of the tag based on the previously entered information.

Simulation modes

The simulator has six different simulation modes. The configured tags are supplied with nearly realistic values during the simulation.

Simulation mode	Description
Sinusoidal	Changes the tag value to form a sinusoidal curve. The value is visualized as a periodic, non-linear function.
Random	Provides randomly generated values. The tag value is changed by means of a random function.
Increment	Increases the value of the tag continuously up to a specified maximum value. Begins again at the minimum after the maximum has been reached. The value trend corresponds to a positive saw-tooth curve.
Decrement	Reduces the value of the tag continuously down to a specified minimum value. Begins again at the maximum after the minimum has been reached. The value curve corresponds to a negative saw-tooth curve.
Shift bit	Shifts a set bit continuously by one position. The previous position is always reset. This lets you test the alarms of an HMI device, for example.
<Display>	The current tag value is displayed statically.

Example: Simulate tags with the "Shift bit" simulation mode

Proceed as follows to simulate tags with the "Shift bit" simulation mode:

1. Open the project you want to simulate.
2. Select the menu command "Online > Simulate Runtime > With tag simulator".

   The tag simulator opens.
3. In the "Tag" column, select a tag from your project.
4. Select "Bin" in the "Format" column.
5. Enter the value "1" in the "Write cycle" column.
6. Select the "Shift bit" simulation mode in the "Simulation" column.
7. Enter the value "1" in the "Set value" column.
8. Enable the tag with the "Start" check box.

Result

The simulator tests the selected tag bit-by-bit as follows:

Simulation values	Byte for alarms
Set start value	00000001
1. Simulation value	00000010
2. Simulation value	00000100
3. Simulation value	00001000
....	...

In Runtime you see if the desired alarm is output at a given value.

See also

Simulating a project (Page 2773)

10.10.3 Servicing the HMI device

10.10.3.1 ProSave

Introduction

The "ProSave" service tool is installed by default when WinCC is installed. The ProSave functions are called in WinCC with the menu "Online > HMI device maintenance".

Functional scope

ProSave provides all of the functions needed to transfer files to the HMI device.

- Data backup and restoration of backed-up data
- Operating system update
- Communication settings

See also

Backup of HMI data (Page 2778)

Updating the operating system (Page 2780)

10.10.3.2 Backup of HMI data

Introduction

Regular backups of the data of an HMI device reduces downtimes to a minimum, e.g. when you replace a device. You simply transfer the backup data to the HMI device and therefore restore the original state.

Data backup with WinCC

If an HMI device is connected to the configuration PC, you can back up and restore HMI device data from the configuration PC using WinCC.

Scope of data backup

Which data is backed up and restored depends on the type of HMI device:

- Complete backup.

  Depending on the HMI device: Runtime, firmware, operating system image, configuration, recipes, user administration, settings
- Recipes only
- User administration only
- Recipes as CSV file

A backup file with the extension *.psb is generated when you backup the data of an HMI device.

As a general rule, you can backup the data to any storage medium. If the HMI device is networked, you can also backup the data to a server.

Note

Scope of data backup

Data backup secures the contents of the flash memory. Alarm logs and process value logs are generally saved on the external storage medium. Alarm logs and process value logs are not backed up. If necessary, back up the contents of the memory card separately.

Note the following when performing a complete data backup and restore operation for HMI devices:

- A full backup includes all options installed.

  As a rule, the backup includes all options data that is still available after "POWER OFF".
- All data previously on the device, including license keys and the operating system, will be permanently deleted when you carry out complete data restoration.
- If the data restoration was interrupted, execute the command "Reset to factory settings". Restart data restoration.

See also

Backing up and restoring data of the HMI device (Page 2779)

ProSave (Page 2777)

10.10.3.3 Backing up and restoring data of the HMI device

Note

Use the restore function for project data only on operating devices which were configured using the same configuration software.

Requirements

- The HMI device is connected to the configuration PC
- The HMI device is selected in the project navigation.
- If a server is used for data backup: The configuration PC has access to the server

Backup of the data of the HMI device

Proceed as follows to backup the data of the HMI device:

1. Select the "Backup" command from the "Online > HMI device maintenance" menu.

   The "SIMATIC ProSave" dialog box opens.
2. Select the data to backup for the HMI device under "Data type".
3. Enter the name of the backup file under "Save as".
4. Click "Start Backup".

This starts the data backup. The backup operation takes some time, depending on the connection selected.

Restoring the data of the HMI device

Proceed as follows to restore the data of the HMI device:

1. Select the "Restore" command from the "Online > HMI device maintenance" menu.
2. Enter the name of the backup file under "Save as".

   Information about the selected backup file is displayed under "Content".
3. Click "Start Restore".

This starts the restoration. This operation takes some time, depending on the connection selected.

Backup/Restore from the "Backup/Restore" dialog in the control panel of the HMI device

The "Backup/Restore" function is enabled for MMC, SD memory cards and USB mass storage devices.

See also

Backup of HMI data (Page 2778)

10.10.3.4 Updating the operating system

Introduction

If the operating system version on an HMI device is incompatible with the configuration, the operating system on the HMI device will be updated automatically (following a prompt) when the project is loaded. Loading will then continue. Loading will otherwise be aborted.

Updating the operating system

Connect the HMI device to the configuration PC to update the operating system of an HMI device. If possible, you should use the interface with the highest bandwidth for this connection, such as Ethernet. Updating the operating system via a serial connection can take up to an hour.

"Reset to factory settings"

If the operating system on the HMI device is no longer operational, update the operating system and reset the HMI device to the factory settings.

Note

You will require the following to reset to the factory settings via Ethernet:

- the MAC address of the HMI device
- the available IP address
- the programming device/PC interface of the configuration PC set to Ethernet TCP/IP

The programming device/PC interface is configured using the control panel of the configuration PC. Select" "S7ONLINE (STEP7) -> TCP/IP" in the "Access point of the application" field.

See also

Updating the HMI device operating system (Page 2781)

ProSave (Page 2777)

10.10.3.5 Updating the HMI device operating system

CAUTION
An operating system update deletes all the data from the HMI device
The operating system update deletes all data from the target system. For this reason, it is advisable to backup the following data: • User administration • Recipes

Requirements

- The HMI device is connected to the configuration PC or the PC with ProSave.
- The HMI device is selected in the project navigation.

Updating the operating system

Proceed as follows to update the operating system:

1. Select the "Update operating system" command from the "Online > HMI device maintenance" menu.
2. Select the "Update operating system" command from the menu under "Online > HMI device maintenance" on the configuration PC in WinCC.

   The "SIMATIC ProSave [OS-Update]" dialog opens. The path to the image of the operating system has been preset.
3. If necessary, select another path to the operating system image you want to transfer to the HMI device.
4. Click "Update OS".

This starts the update. The update operation can take time, depending on the connection selected.

Resetting the HMI device to factory settings

To reset the HMI device to factory settings, proceed as follows:

1. Switch off power to the HMI device.
2. Select the "Update operating system" command from the menu under "Online > HMI device maintenance" on the configuration PC in WinCC.

   The "SIMATIC ProSave [OS-Update]" dialog opens. The path to the image of the operating system has been preset.
3. If necessary, select another path to the operating system image you want to transfer to the HMI device.
4. Activate "Reset to factory settings".
5. Click "Update OS".
6. To "Reset to factory settings", switch on power to the HMI device again.

This operation can take time.

Result

The operating system of the HMI device is updated to the latest version.

See also

Updating the operating system (Page 2780)

10.10.4 Reference

10.10.4.1 Error messages during loading of projects

Possible problems during the download

When a project is being downloaded to the HMI device, status messages regarding the download progress are displayed in the output window.

Usually, problems arising during the download of the project to the HMI device are caused by one of the following errors:

- Wrong operating system version on the HMI device
- Incorrect download settings on the HMI device
- Incorrect HMI device type in the project
- The HMI device is not connected to the configuration PC.

The most common download failures and possible causes and remedies are listed below.

The serial download is cancelled

Possible remedy: Select a lower baud rate.

The download is cancelled due to a compatibility conflict

Possible cause	Remedy
Conflict between versions of the configuration software and the operating system of the HMI device	Synchronize the operating system of the HMI device with the version of the configuration software. To update the operating system on the HMI device, select the "Update operating system" command from the "Online > HMI device maintenance" menu in WinCC. You can also use ProSave. For additional information, refer to the operating instructions for the HMI device.
The configuration PC is connected to the wrong device, e.g. a controller.	Check the cabling. Correct the communication parameters.

Project download fails

Possible cause	Remedy
Connection to the HMI device cannot be established (alarm in the output window)	Check the physical connection between the configuration PC and the HMI device. Check whether the HMI device is in transfer mode. Exception: Remote control
The default communication driver is not listed in the Windows Device Manager.	Check the device status of the COM connection in the properties window of the Device Manager.

Download over MPI/DP interface fails

Possible cause	Remedy
"Configured mode" is set on the CP, for example, if you are using the SIMATIC NET CD.	Set the CP to "PG mode" using the "Set PC station" application. Check the "baud rate" and "MPI address" network parameters. Download the project from WinCC to the CP. Set the CP back to "configured mode".
On the programming device/PC panel, the "S7ONLINE" access point is not set to a hardware device such as CP5611 (MPI). The cause may be the installation of "SIMATIC NET CD 7/2001".	Set the access point "S7ONLINE" on the selected device using the "PG/PC Panel" or "Set PC station" application. Check the "baud rate" and "MPI address" network parameters. Download the project from WinCC to the HMI device. Restore the "S7ONLINE" access point to the original device.

The configuration is too complex

Possible cause	Remedy
The configuration contains too many different objects or options for the HMI device selected.	Remove all objects of a type, e.g. all graphic views.

10.10.4.2 Adapting the project for another HMI device

Introduction

When you download a WinCC project to an HMI device, WinCC checks whether this is compatible with the HMI device type used in the project. If the types of HMI device do not match, you will see a message before the download starts.

The download is aborted.

Adapting the project for the HMI device

You need to adapt the project accordingly to be able to download the project to the connected HMI device.

- Add a new HMI device in the project tree. Select the correct type of HMI device from the HMI device selection.
- Copy the configured components from the previous to the new HMI device.

  Copy large amounts of components directly in the project navigation and details view.

  For example, copy the "Screens" folder to the screens folder of the new HMI device with the help of the shortcut menu.

- Use the detail view to copy entries in the project tree for which the "Copy" command is not available in the shortcut menu.
- Select the "Recipes" entry in the project tree, for example. The recipes are displayed in the detail view.
- Select the recipes in the detail view and drag them to the "Recipes" entry of the new HMI device. The recipes are copied. You can also select multiple objects in the detail view.
- Configure the components that cannot be copied, e.g. connections, area pointers, and alarms.
- Save the project at various points in time.
- Compile the full project.
- When the compilation is successfully completed, download the project to the HMI device.

Linking references

References to linked objects are included in the copying. The references are linked again once the linked objects are copied.

Example:

You copy a screen in which objects are linked to tags. The tag names are entered at the individual objects after the screen is added to the new HMI device. The tag names are marked in red because the references are open. When you then copy the tags and insert them into the new HMI device, the open references are closed. The red marking for the tag names disappears.

For complete references to connected objects in the PLC, you first need to configure a connection to the PLC.

Using the information area

When you compile the project for the HMI device, errors and warnings are displayed in the "Info" tab of the Inspector window. You can use the shortcut menu command "Go to" to go directly to the location where the error or warning can be corrected.

Work through the list of errors and warnings from top to bottom.

When the compilation is successfully completed, download the project to the HMI device.

10.10.4.3 Establishing a connection to the HMI device

Introduction

To download a WinCC project to an HMI device, a properly configured connection must be set up between the configuration PC and HMI device. The connection cannot be set up, the download is cancelled.

Setting up a connection between the configuration PC and HMI device

1. Check the cable connection between the HMI device and configuration PC.
2. Open the "Devices & Networks" editor in WinCC and start the network view.
3. Select the subnet in the network view and check the settings for the subnet.
4. Select the interface of the HMI device in the network view or device view and check the connection parameters in the Inspector window.
5. Switch on the HMI device and press the "Control Panel" button in the loader.

   The Control Panel opens.
6. Press "Transfer" twice in the Control Panel.

   The "Transfer Settings" dialog box opens.
7. Check the settings and then press "Advanced".

   The [Protocol*] Settings" dialog opens.

   *: The title of the dialog depends on the protocol used, for example, "PROFIBUS Settings".
8. Check the advanced settings and close the dialog with "OK".

Important settings

Check the connection settings and in particular the following parameters:

- Network and station addresses
- Selected transmission rate
- Master on the bus; as a general rule, only one master is permitted.

If using a configurable adapter for the connection, check the adapter settings, for example, transmission rate, master on the bus.

10.11 Operating in Runtime

10.11.1 Basics

10.11.1.1 Overview

Configuration and process control phases

HMI devices are used to operate and monitor tasks in process and production automation. The plant screens on the HMI devices provide a clear overview of active processes. The HMI device project, which includes the plant screens, is created during the configuration phase.

Transfer the project to the HMI device for the process control phase. Another requirement for the process control phase is that the HMI device must be connected online to a PLC. The process control phase, operator control, and monitoring can then be carried out during an ongoing work process.

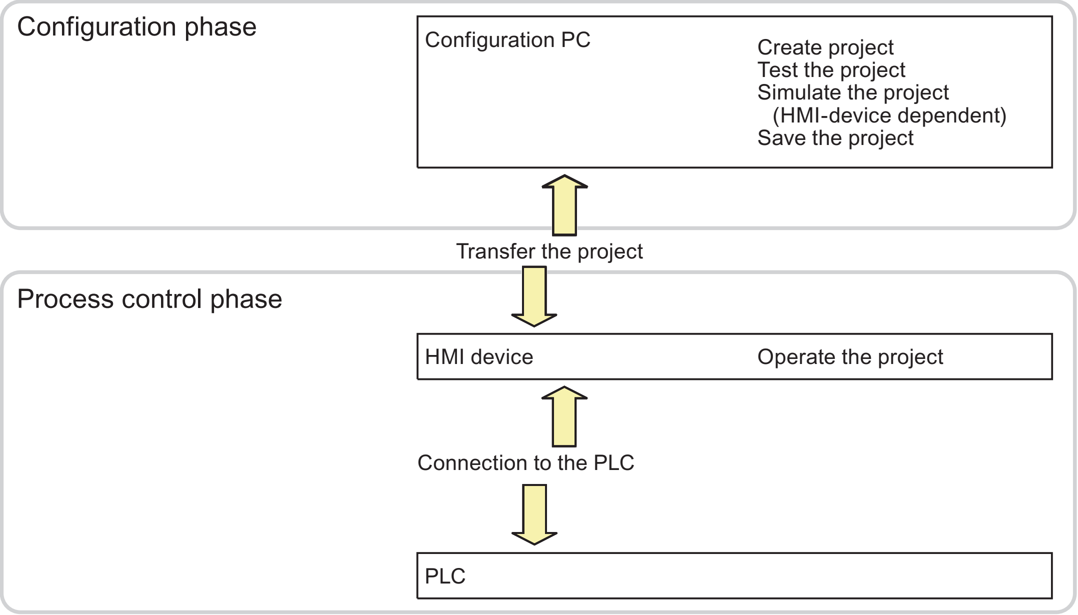


Figure 10-1 Configuration and process control phases

Downloading the project to the HMI device

The following procedures are available to download a project to an HMI device:

- Downloading from the configuration PC
- Restore the project from a PC using ProSave

  In this case, an archived project is downloaded from a PC to the HMI device. The configuration software need not be installed on this PC.

These procedures are available for commissioning and recommissioning a project.

Commissioning and Recommissioning

- When the HMI device is commissioned there is no project at first.

  The HMI device is also in this state after the operating system has been updated.
- When recommissioning, any project on the HMI device is replaced.

10.11.1.2 Tags in Runtime

Definition

Tags correspond to defined memory areas on the HMI device, to which values are written and/or from which values are read. This action can be initiated by the PLC, or by the operator at the HMI device.

10.11.1.3 System functions in Runtime

Application

System functions are used in Runtime for the following purposes:

- To control the process.
- To utilize the properties of the HMI device.
- To configure system setting on the HMI device in online mode.

Each system function in WinCC is logically linked with an object and an event. As soon as the event occurs, the system function is triggered.

System functions

These default system functions are used to implement many tasks in Runtime, such as:

- Calculations, e.g. increasing a tag value by a specific or variable amount.
- Logging functions, e.g. starting a process value log.
- Settings, e.g. PLC changes, or setting a bit in the PLC.
- Alarms, e.g after a different user logs on.

Events

The object and the selected function determine the event that can be defined as a trigger for executing a system function.

For example, the "Change value", "Low limit violated" and "Upper limit exceeded" events are associated with the "Tag" object. The "Loaded" and "Cleared" events are associated with the "Screen" object.

10.11.2 Commissioning projects

10.11.2.1 Settings in the Runtime software

Open the WinCC configuration software and make the following settings for the runtime software:

Display on the PLC

In WinCC, configure the visual representation of the generated project in runtime. The screen resolution is fixed for Basic Panels. Scroll bars will appear if the screen is larger than the configured screen resolution.

To switch off the taskbar, select the Start menu command "Settings > Taskbar". In the "Properties of the Taskbar" dialog, disable the options "Always on top" and "Auto hide".

Dialog fonts

The dialog text will be shown in the standard font. Define the default font under "Language & Font" in the "Device Settings" editor.

Disabling program switching

You lock program switching to prevent operators from calling other applications in Runtime.

Click "Window" in the "Device Settings" editor and select the option "Disable program switching". Also hide the Windows taskbar.

Note

Stop runtime

Always configure a softkey or button for calling the "StopRuntime" system function if you lock program switching. Otherwise, you will not be able to close either runtime or Windows.

Screen saver

A screensaver is no longer required for most modern screens and can, in fact, even cause damage. These monitors switch to hibernate mode as soon as the video signal has not changed for a specified time. A conventional screensaver would prevent this and thus reduce the service life of your monitor.

Note

Approved screensavers

If you do want to use a screensaver, note that only the standard Windows screensavers are approved for use in Runtime.

Make sure that the correct time zone is set on the PC on which the runtime software is installed. To set the time zone in Windows, select Start > Settings > Control Panel > Date and Time.

10.11.2.2 Loading projects

Overview

Various scenarios are possible for loading the project:

- The Runtime software is installed on the same system as the configuration software.
- The Runtime software and the configuration software are installed on different systems. The project must be loaded from the configuration computer to the target system. The HMI devices must be connected to the configuration PC for the transfer. Another requirement is that the transfer mode must match on the HMI devices and in WinCC.

  Note

  Security prompts may appear during the loading process, depending on the configuration. The recipe data and password list on the HMI device are overwritten following a prompt.

The configuration software and the Runtime software are installed on the same system

If the configuration software and the Runtime software are installed on the same system, proceed as follows:

1. Create and compile your project.
2. Start Runtime directly from the active configuration software. Select the "Start Runtime" command from the "Online" menu.
3. You may test and operate the project online with the controller if you have configured the corresponding communication.

The configuration software and the Runtime software are installed on different systems

If the configuration software and the Runtime software are installed on different systems, proceed as follows:

1. Create and compile your project. For additional information, refer to "Compiling a project".
2. To download the file via cable:

   Connect the HMI device to the configuration computer using a standard cable to match the desired transfer mode and then switch on the HMI device.
3. Set the HMI device to transfer mode.

   To start transfer mode, press the "Transfer" button in the Loader. You can also assign the system function "SetDeviceMode" to an operator control.
4. Load the project from the configuration computer to your target device. For further information, refer to "Loading projects".

Note

If the HMI device is a PC, you can transfer the compiled file without using the loader, for example, via Ethernet. Double-click the corresponding file on your PC to start Runtime.

10.11.2.3 Starting Runtime on the Engineering Station

Introduction

You can start your project in Runtime at the engineering station while performing the configuration in WinCC. However, this so-called online configuration is subject to certain limitations.

The project cannot be compiled in the background while Runtime is active on the Engineering Station. The delta data of the project is compiled automatically when you load the project to an HMI device after having closed Runtime. You can also start compilation manually.

When the project is started in Runtime, the settings you have stored in your project for the HMI device in the "Configuration" editor take effect.

Requirement

A project is open on the Engineering Station.

Procedure

Proceed as follows to start Runtime on the Engineering Station:

1. Select the desired HMI device in the project tree.
2. Select the menu command "Online > Start Runtime".
3. If you want to edit project data after Runtime was started on the Engineering Station, select the "Compile > Software" command in the shortcut menu of the HMI device.

The updated project is displayed in the Runtime on the Engineering Station.

10.11.2.4 Starting Runtime on the HMI device

Introduction

On completion of the project download to the HMI device, you can start the project in Runtime. The project is saved in the HMI device to a file with the following extension:

- Basic Panels as well as OP 73, OP 77A and TP 177A: "*.srt"
- All other HMI devices: "*.fwc"

The project settings defined in the "Runtime settings" of the HMI device are activated when the project is started in Runtime.

The programs that you can use to start projects on the HMI device are available in the Runtime installation folder.

Requirements

WinCC Runtime is installed on the HMI device.

Procedure

The "RT Loader" application is started on a panel. The project loaded is started automatically after expiration of the configured delay.

If the project does not start automatically:

1. To select the project file, click "Settings" and enter the path to the project file under "Configuration file".
2. Click "OK" and then "Start".

10.11.2.5 Testing a project

Introduction

You have the following options for testing a WinCC project:

Testing projects on the configuration computer.

- Simulator

  The Simulator is used to test WinCC projects with internal tags and process tags. For additional information, refer to "Simulating a project".

  The simulator enables you to test the following:

  – Offline testing of a configuration without connection to a PLC.

  – Online testing of a configuration with connection to a PLC and inactive process.

  – Implementation of a demo project.

Testing projects on the HMI device

- Offline testing of the project on the HMI device

  Offline testing means that communication between the HMI device and the PLC is down for the duration of the test. You can operate the HMI device, but you cannot exchange data with the PLC and vice versa. Set the "Offline" mode on the HMI device by assigning the system function "SetDeviceMode" to an operator control.
- Online testing of the project on the HMI device

  Online testing means that communication between the HMI device and the PLC is up for the duration of the test. You control the plant using the HMI device based on the configuration. Set the "Online" mode on the HMI device by assigning the system function "SetDeviceMode" to an operator control.

Procedure

Proceed as follows to simulate a project without a PLC connection to the configuration PC:

1. Create a project as it is going to be run later with an interconnected PLC.
2. Save and compile the project.
3. Start the Simulator directly from the active configuration software. Select the menu command "Online > Simulate Runtime > With tag simulator".

   When you simulate the project for the first time, the simulator is started with a new, empty simulation table. If you have already created a simulation table for your project, it is opened.

   The simulation table "*.six" contains all the settings required for the simulation. For additional information, refer to "Working with the tag simulator".
4. Make any changes to the tags and area pointers of your project in the simulation table.

   Toggle between the simulation table and Runtime using the <ALT+TAB> key shortcut.

   Save the settings for the simulation using the menu command "File > Save." Enter a name to save the file.. The file name is automatically assigned the extension "*.six".

10.11.2.6 Closing a project

Introduction

You define the steps in closing Runtime in the user program:

Procedure

Exit Runtime as follows:

1. When Runtime is running, you can close it using the close symbol or the Task Manager.
2. When Runtime is running, press the relevant button to close Runtime. The close of Runtime is especially configured.

10.11.2.7 Backing up and restoring data of the HMI device

Introduction

Backup the data of an HMI device at regular intervals.

Working from the engineering station to which an HMI device is connected, you can backup and restore the data of this HMI device using WinCC.

You have the option of conveniently performing a central data backup using ProSave on a computer without WinCC installation.

Requirement

- The HMI device is interconnected with the Engineering Station or the computer is connected with ProSave.
- The HMI device whose data should be backed up or restored is selected in the project tree.
- The settings for loading are correctly set in the properties of the HMI device.
- When using a special storage medium, such as a data server: The HMI device is connected to the storage medium.

Procedure

Proceed as follows to back up the data:

1. Select the "Backup" command from the "Online > Device maintenance" menu.
2. Select the scope of the backup: "Complete backup," "Recipes," or "User administration."
3. Click "...", select the storage location in the "Select backup file" dialog, and specify a file name.
4. Click "OK."

This starts the data backup. The backup can take some time.

Procedure for restoring data

1. Select the "Restore" command from the "Online > Device maintenance" menu.
2. Click "...", and select the storage location and file in the "Open" dialog.

   The "Content" area indicates which HMI device is the origin of the data backup and the scope of the data backup.
3. Click "OK."

This starts the restoration. This process can take some time.

10.11.3 Languages in runtime

10.11.3.1 Languages in runtime

Using multiple runtime languages

You can decide which project languages are to be used in runtime on a particular HMI device. The number of runtime languages that can be available at one time on the HMI device depends on the device. To enable the operator to switch between languages during runtime, you must configure a corresponding operator control.

When runtime starts, the project is displayed according to the most recent language setting. When runtime starts the first time, the language with the lowest number in the "Language order" is displayed.

Setting runtime languages during configuration

You can specify the following in the "Language and Font" editor:

- The project languages available as runtime languages for the HMI device.
- The order in which the languages are switched.

10.11.3.2 Setting a runtime language

Introduction

The "Language & Font" editor shows all project languages available in the project. You can select the project languages to be available as runtime languages on the HMI device. In addition, you specify the order in which the languages will be switched.

Requirement

Multiple languages are enabled in the "Project languages" editor.

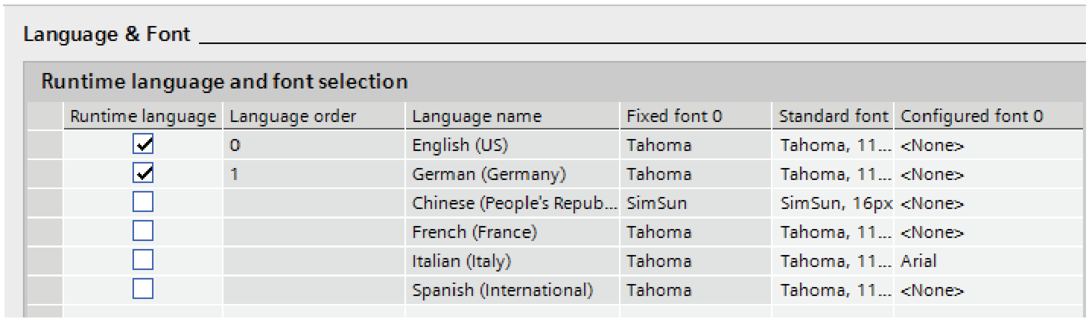
Procedure

1. Open the "Language & Font" editor under "Device settings".
2. In the "Runtime language" column, select the the languages to be used when runtime first starts.

   The selected language is assigned the number "0" in the "Language order" column.

3. In the "Runtime language" column, select the language to be activated next when the language is switched.

   The selected language is assigned the number "1" in the "Language order" column.



Language & Font

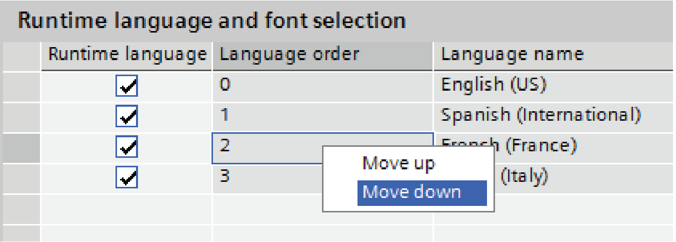
Runtime language and font selection

Runtime language	Language order	Language name	Fixed font 0	Standard font	Configured font 0
☑	0	English (US)	Tahoma	Tahoma, 11...	<None>
☑	1	German (Germany)	Tahoma	Tahoma, 11...	<None>
☐		Chinese (People's Repub...	SimSun	SimSun, 16px	<None>
☐		French (France)	Tahoma	Tahoma, 11...	<None>
☐		Italian (Italy)	Tahoma	Tahoma, 11...	Arial
☐		Spanish (International)	Tahoma	Tahoma, 11...	<None>

4. Select additional languages in the order in which they are to be activated when the language is switched.

   If the number of languages selected exceeds the number that can be loaded on the HMI device, the table background changes in color.

5. If you want to change the order of a language, select the desired row and then select the shortcut menu command "Move up" or "Move down".



Runtime language and font selection

Runtime language	Language order	Language name
☑	0	English (US)
☑	1	Spanish (International)
☑	2	French (France)
☑	3	(Italy)

Result

The enabled runtime languages are transferred with the compiled project to the HMI device.

When runtime starts the first time, the project is displayed in the language with the lowest number in the "Language order."

If language switching by means of the "SetLanguage" system function has been configured, the specified number sequence determines the order in which the languages are switched.

10.11.3.3 Setting the font for a runtime language

Introduction

You can specify the font used to display the texts for each runtime language on the HMI device in the "Language & Font" editor. The default font is used in all texts if you cannot define a specific font.

WinCC offers only fonts supported by the HMI device.

The default font is also used for the representation of dialogs in the operating system of the HMI device. Select a smaller font as default if the full length of the dialog texts or headers is not displayed.

Requirement

Multiple languages are enabled in the "Project languages" editor.

Procedure

1. Open the "Language & Font" editor under "Device settings".
2. Enable the languages to be displayed on the HMI device in the "Runtime language" column.

   In the "Fixed font 0" column, WinCC shows the fonts used by default in runtime.
3. In the "Configured font 0" column, select another font for each language you want to have available during configuration.

   These fonts are transferred to the HMI device during a transfer operation.
4. In the "Standard font" column, select the font to be used by default if a font cannot be selected for a text.

Result

The project texts for the selected language are displayed in the selected font on the HMI device.

10.11.3.4 Configuring language switching

Introduction

You need to configure language switching if you want to have multiple runtime languages available on the HMI device. This is necessary to enable the operator to switch between the various runtime languages.

Methods for language switching

You can configure the following methods for language switching:

- Direct language selection

  Each language is set by means of a separate button. In this case, you create a button for each runtime language. Configure the "SetLanguage" system function for each font with the number of the language or the language ID as a parameter.
- Language switching

  The operator toggles through all languages using a button.

  Configure "SetLanguage" system function for the button with the "Toggle" parameter. The language is enabled in the order you have specified in the "Language & Font" editor.

Regardless of the method used, the button names must be translated into each of the languages used. You can also configure an output field that displays the current language setting.

10.11.3.5 Specific features of Asian and Eastern languages in runtime

Introduction

When configuring for Asian languages some specific features should be observed for operation in runtime.

Note

You can only use Asian fonts supported by your configuration computer during configuration.

Memory requirement for Asian character sets

The memory requirement is of course greater when using Asian languages. Therefore, pay attention to any error messages when compiling the project.

Inputting Eastern and Asian characters (not ANSI)

You cannot enter Eastern and Asian characters in runtime on the Basic Panels.

Interpretation of Asian characters

When using Sm@rtAcess and Sm@rtService, only the characters known on the HMI device can be used. To be able to use Asian characters, these must be configured in the engineering system. Additionally configured characters require additional memory on the HMI device. Pay attention to the amount of available memory on the HMI device.

Font size for Asian character sets

Use at least a font size of 10 points to display the text of projects created for Asian languages in runtime. Asian characters will become illegible if smaller font sizes are used. This also applies to the default font used in the "Language & Font" editor.

Text field length for Asian languages

Make allowances for an appropriate length of the text fields when working on multilingual projects with Asian languages. Field contents may be partially hidden, depending on the font and the font size.

1. Open the "Properties > Appearance" text box in the Inspector window.
2. Under "Fit to size", disable the "Auto-size" option.
3. Verify the proper display in runtime.

10.11.4 Operating projects

10.11.4.1 Basics

Overview of operator control over a project

All Basic HMI devices feature a touch screen. Certain Basic HMI devices feature function keys. Use the touch screen and function keys to operate the Control Panel or the project running on your HMI device.

⚠ DANGER

Incorrect operation

A project can contain certain operations that require in-depth knowledge about the specific plant on the part of the operator.

Ensure that only trained professional personnel operate the plant.

Operating the touch screen

CAUTION

Damage to the touch screen

Pointed or sharp objects can damage the plastic surface of the touch screen.

Always operate the touch screen with your fingers or with a touch pen only.

Triggering unintended actions

Touching several operator controls at the same time can trigger unintended actions.

Touch only one operator control on the screen at a time.

Operator controls are touch-sensitive symbols on the screen of the HMI device.

They are basically operated in the same way as mechanical keys. You activate operator controls by touching them with your finger.

Note

The HMI device returns a visual feedback as soon as it detects that an operator control has been touched.

The visual feedback is independent of any communication with the PLC. The visual feedback signal therefore does not indicate whether or not the relevant action is actually executed.

Examples of operator controls:

- Buttons

  Buttons can assume the following states:

"Untouched"

"Touched"

- Invisible buttons

  The focus of invisible buttons is by default not indicated following selection. No optical operation feedback is provided in this case.

  The configuration engineer may, however, configure invisible buttons so that their outline appears as lines when touched. This outline remains visible until you select another operator control.

- I/O fields

  A screen keyboard appears as visual feedback after you touched an I/O field, for example, to enter a password.

  Depending on the HMI device and the configured operator control, the system displays different screen keyboards for entering numerical or alphanumerical values.

  The screen keyboard is automatically hidden again when input is complete.

Operating function keys

The function keys can be assigned global or local functions:

- Function keys with global function assignment

  A function key with global function assignment always triggers the same action on the HMI device or in the PLC, regardless of the currently displayed screen. The activation of a screen or the closing an alarm window, for example, is such an action.

- Function keys with local function assignment

  A function key with local function assignment is screen-specific and is therefore only effective within the active screen.

  The function assigned to a function key can vary from screen to screen.

The function key could be assigned only a single function within a screen only, that is, either a global or a local function. Local function assignments override global function assignments.

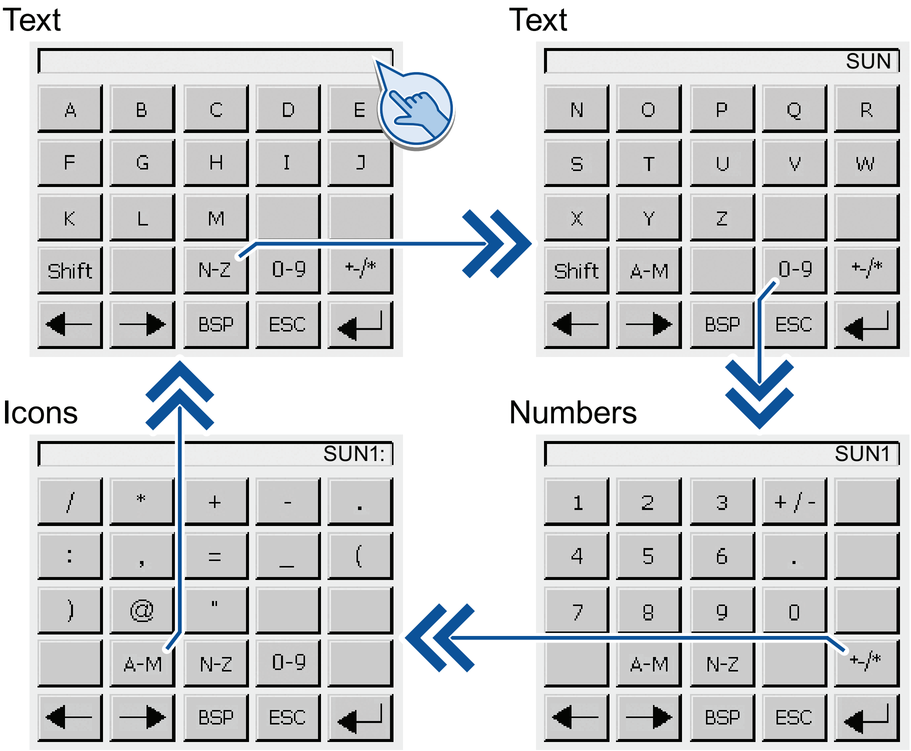
General functions of the screen keyboard

The following keys are available on the screen keyboard of all Basic HMI devices:

←	Cursor left
→	Cursor right
BSP	Deleting a character
ESC	Cancel input
↵	Confirm input
Help	Displaying infotext. This key only appears when an infotext has been configured for the operator control.

Entering data on the KTP400 Basic

Due to the small display, the screen keyboard and the input concept of the KTP400 Basic differs compared to other Basic HMI devices.



The screen keyboard appears on the HMI device touch screen when you touch an operator control that requires input.

The screen keyboard of the KTP400 features four views. You can change the view while making entries using the buttons in the fourth row of the screen keyboard:

Button	Changes to the view
A-M	Entering text, characters "A" to "M"
N-Z	Entering text, characters "N" to "Z"
0-9	Entering numbers, "0" to "9," signed or unsigned and with or without decimal places
+-/*	Entering special characters
Shift	Entering text, shift to lower case letters

Note

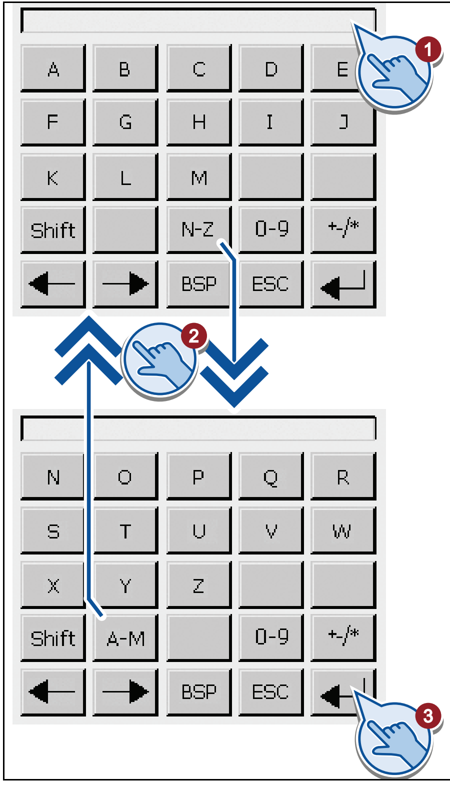
Job mailbox has no effect

PLC job 51 "Select screen" has no effect while the screen keyboard is open.

Key assignment

The alphanumerical keyboard layout is monolingual.
A language change within the project does not have any effect on the layout of the alphanumerical screen keyboard.

Entering alphanumerical values



1. Touch the desired operator control on the screen.

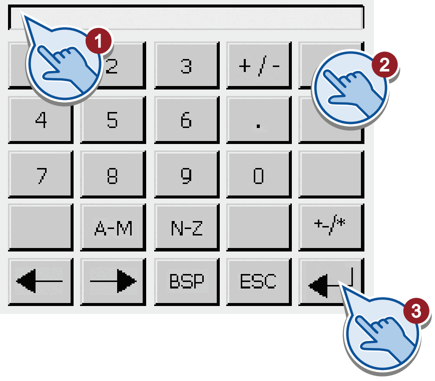
   The alphanumerical screen keyboard opens.
2. Enter the value. Depending on the settings, the HMI device outputs an audible signal.

   You can change the view of the screen keyboard using the keys <N-Z> and <A-M>.

   Use the <Shift> key to enter lower-case letters.
3. Press <Return> key to confirm your entries, or cancel them with <ESC>.

   Either action closes the screen keyboard.

Entering numerical values



1. Touch the desired operator control on the screen.

   The numerical screen keyboard opens.
2. Enter the value. Depending on the settings, the HMI device outputs an audible signal.

   You can change the view of the screen keyboard for entering numbers with hexadecimal notation using the <N-Z> and <A-M> keys.
3. Press <Return> key to confirm your entries, or cancel them with <ESC>.

   Either action closes the screen keyboard.

Checking numerical value limits

Tags can be assigned limit values. Any entry of a value outside this limit is rejected. If an alarm view is configured, a system event is triggered and the original value is displayed again.

Decimal places of numerical values

The configuration engineer can define the number of decimal places for a numerical text box. The number of decimal places is checked when you enter a value in this type of I/O field.

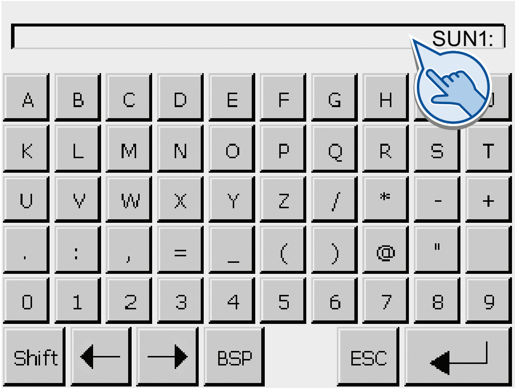
- Decimal places that exceed the limit are ignored.
- Unused decimal places are padded with "0" entries.

Entering data on the KTP600, KTP1000, TP1500 Basic

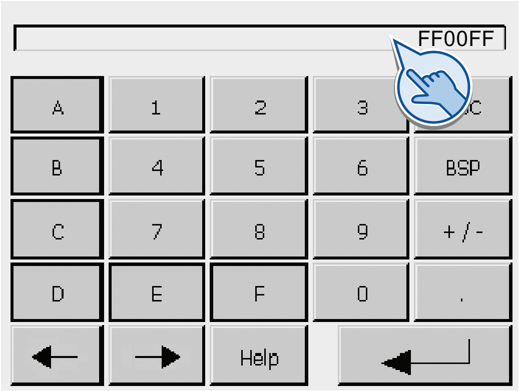
Alphanumerical screen keyboard

The screen keyboard appears on the HMI device touch screen when you touch an operator control that requires input.

Text



Numbers



Note

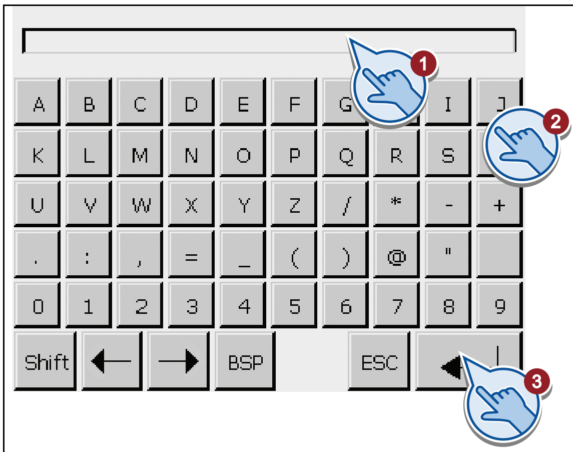
Job mailbox has no effect

PLC job 51 "Select screen" has no effect while the screen keyboard is open.

Key assignment

The alphanumerical keyboard layout is monolingual.
A language change within the project does not have any effect on the layout of the alphanumerical screen keyboard.

Entering alphanumerical values



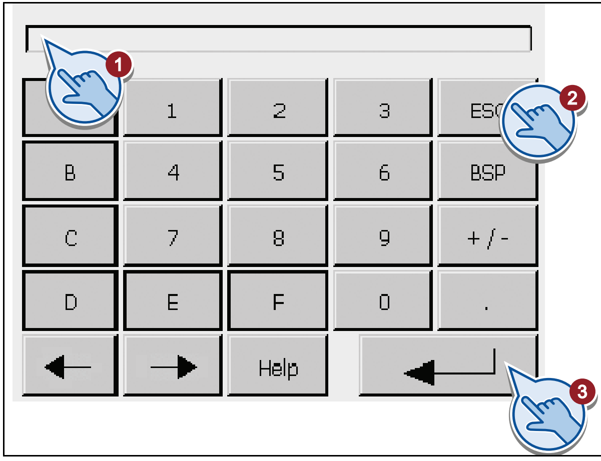
1. Touch the desired operator control on the screen.

   The alphanumerical screen keyboard opens.
2. Enter the value. Depending on the settings, the HMI device outputs an audible signal.

   Use the <Shift> key to enter lower-case letters.
3. Press <Return> key to confirm your entries, or cancel them with <ESC>.

   Either action closes the screen keyboard.

Entering numerical values



1. Touch the desired operator control on the screen.

   The numerical screen keyboard opens.
2. Enter the value. Depending on the settings, the HMI device outputs an audible signal.
3. Press <Return> key to confirm your entries, or cancel them with <ESC>.

   Either action closes the screen keyboard.

Checking numerical value limits

Tags can be assigned limit values. Any entry of a value outside this limit is rejected. If an alarm view is configured, a system event is triggered and the original value is displayed again.

Decimal places of numerical values

The configuration engineer can define the number of decimal places for a numerical text box. The number of decimal places is checked when you enter a value in this type of I/O field.

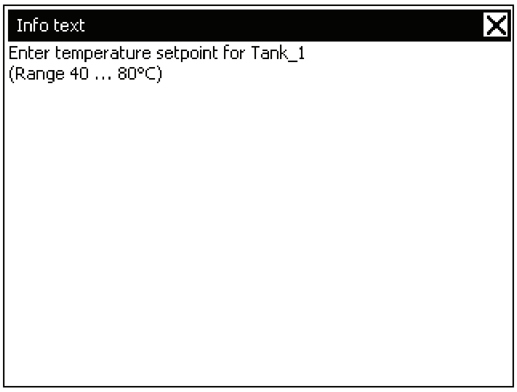
- Decimal places that exceed the limit are ignored.
- Unused decimal places are padded with "0" entries.

Display tooltip

Use

The configuration engineer uses tooltips to provide additional information and operating instructions. The configuration engineer can configure tooltips for screens and operating elements.

The tooltip of an I/O field may contain, for example, information on the value to be entered.



Enter temperature setpoint for Tank_1
(Range 40 ... 80°C)

Procedure

Proceed as follows to open the tooltip for operator controls:

1. Touch the required operating element.

   The screen keyboard opens. The representation of the Help key indicates whether a tooltip was configured for the operating element or for the current screen.
2. Touch the Help key of the on-screen keyboard.

   The tooltip for the operating element is displayed. If there is no tooltip for the selected screen object, the tooltip for the current screen is displayed if it has been configured.

   You can scroll through the contents of long tooltips using the ▼ and ▲ buttons.

   Note

   Switching between displayed tooltips

   The configuration engineer can configure tooltips for an I/O field and the associated screen. You can switch between two tooltips by touching the tooltip window.
3. Close the displayed tooltip by pressing ☒.

Alternative procedure

Depending on your configuration, tooltips can also be called via a configured operating element.

Setting the project language

Introduction

The HMI device supports multilingual projects. You must have configured a corresponding operating element which lets you change the language setting on the HMI device during runtime.

The project always starts with the language set in the previous session.

Requirement

- The required language for the project must be available on the HMI device.
- The language switching function must be logically linked to a configured operating element such as a button.

Selecting a language

You can change project languages at any time. Language-specific objects are immediately output to the screen in the new language when you switch languages.

The following options are available for switching the language:

- A configured operating element switches from one language to the next in a list.
- A configured operating element directly sets the desired language.

10.11.4.2 Operating objects

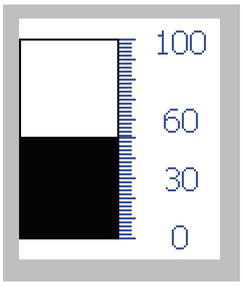
Bar

Application

The bar is a dynamic display object. The bar displays a value from the PLC as a rectangular area. The bar allows you to recognize the following at a glance:

- The distance of the current value from the configured limit values
- Whether a set point value has been reached

The bar can display values such as fill levels or batch counts.



Layout

The layout of the bar depends on the configuration:

- The bar may feature a scale of values
- The configured limit values can be indicated by lines
- Color changes can signal when a limit value has been exceeded or has not been reached

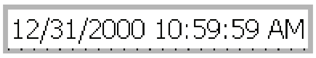
Date/time field

Overview

Application

A "Date / time box" may have the following runtime functions:

- Output of the date and time
- Combined input and output; here the operator can edit the output values so as to reset the date and time.



12/31/2000 10:59:59 AM

Layout

The appearance of the date/time field depends on the language set in the HMI device.

The date can be displayed in detail (e.g. Tuesday, 31 December 2003) or in short form (31.12.2003).

Operation

Depending on the configuration, you can operate the date/time field in the following ways:

- Standard operation: Change date and time.

Runtime behavior

When the operator ignores the syntax when entering values, or enters illegal values, the system rejects these. Instead, the original values (plus the time that has elapsed in the meantime) appears in the date/time field and a system alarm is displayed on the HMI device.

Touch and key operation

Touch operation

Proceed as follows to operate the date/time field:

1. Touch the date/time field on the touch screen of the HMI device. The on-screen keyboard is displayed automatically.
2. Enter the required value using the on-screen keyboard.
3. Press <ENTER> to confirm your entry, or cancel it with <ESC>. The on-screen keyboard is cleared automatically after you have confirmed or canceled your entries.

Key operation

Activate the date/time field, for example, with one or several TAB according to the configured tab sequence. A color frame signals the selected state of the field content.

Procedure

Proceed as follows to operate the date/time field:

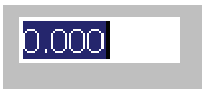
1. Position the cursor using the cursor keys and enter the value.
2. PressENTER. The object changes to the special editing mode. Only one character at any time is marked in the field.
   - Use the cursor keys ▲ / ▼ to scroll a character table.
   - Use the cursor keys ◀/▶ to move the cursor to the next or previous input position.
3. Confirm your entry with ENTER, or discard it with ESC.

I/O field

Overview

Application

You enter numeric or alphanumeric values in an I/O field. For example, a numeric value could be the number 80 as a temperature reference; an alphanumeric value could be the text "Service" as a user name.


0.000

Layout

The appearance of the I/O field depends on the configuration:

- Numeric I/O field

  For input of numbers in decimal, hexadecimal or binary format.
- Alphanumeric I/O field

  For input of character strings.
- I/O field for date and time

  For input of calendar dates or time information. The format depends on the set configuration.
- I/O field for password entry

  For concealed entry of a password. The entered character string is displayed with placeholders (*).

Operation

Depending on the configuration, you can operate the I/O field in the following ways:

- Standard operation: Enter a value in the I/O field.
- Event: An event is triggered when you operate the I/O field, for example, when you activate it. The processing of a function list can be configured to the event.

Runtime behavior

Limit value test of numerical values

Tags can be assigned limit values. If you enter a value that lies outside of this limit, it will not be accepted; for example, 80 with a limit value of 78. In this case the HMI device will deliver a system alarm, if an alarm window is configured. The original value is displayed again.

Decimal places for numerical values

The configuration engineer can define the number of decimal places for a numerical text box. The number of decimal places is checked when you enter a value in this type of I/O field.

- Decimal places in excess of the limit are ignored.
- Empty decimal places are filled with "0".

Hidden input

A "*" is displayed for every character during hidden input. The data format of the value entered cannot be recognized.

Behavior when switching between input fields

When you change to another input field within the same screen, and the screen keyboard appears, the "Exit field" event is not executed for the previous field unless you close the screen keyboard.

Touch and key operation

Touch operation

Proceed as follows to operate the IO field:

1. Touch the IO field on the touch screen of the HMI device. The on-screen keyboard is displayed automatically.
2. Enter the required value using the on-screen keyboard.
3. Press <ENTER> to confirm your entry, or cancel it with <ESC>.

The on-screen keyboard is cleared automatically after you have confirmed or canceled your entries.

Key operation

Activate the IO field,for example, with one or several TAB according to the tab sequence configured. A color frame signals the selection of the field content.

Procedure

Proceed as follows to operate the IO field:

1. Position the cursor using SHIFT and a cursor key.
2. This step cancels the field content selection. Enter the desired value.
3. PressENTER. The object will change to special editing mode. Only one character at any time is marked in the field.
   - Use the cursor keys ▲ / ▼ to scroll a character table.
   - Use the cursor keys ▶ / ◀ to move the cursor to the next or previous input position.
4. Confirm your entry with ENTER, or discard it with ESC.

Note

Enable the input of hexadecimal characters o "F" for numerical values by toggling the input keys to character mode using the key A-Z.

Graphic view

Application

The graphic view displays graphics.

Layout

The appearance of the graphic depends on the configuration. The graphic view, for example, adapts automatically to the size of the graphic.

Note

If you use bitmaps with the "Transparent color" setting in WinCC screens, requires high-performance display on the Panel HMI devices. Performance is enhanced by disabling the "Transparent color" setting in the properties of the respective graphic object. This restriction applies in particular when bitmaps are used as background image.

Operation

The graphic display is for display only and cannot be operated.

Graphic I/O field

Overview

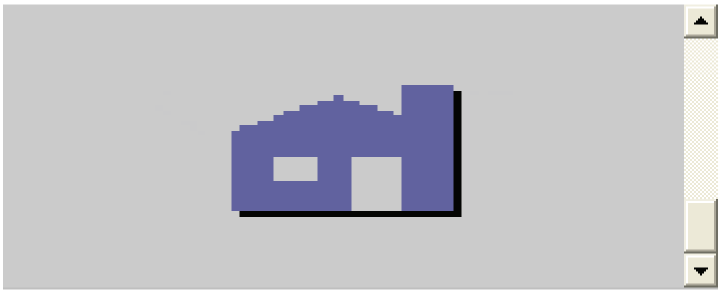
Application

A graphic IO field can have the following Runtime functions:

- Output of graphic list entries
- Combined input and output

Example of its use as output field:

To indicate the Runtime status of a valve, the graphic IO field outputs the image of a closed or open valve.



Operation

Depending on the configuration, you can operate the graphic IO field in the following ways:

- Standard operation: Select an entry from the graphic list.
- Event: An event is triggered when you operate the graphic IO field, e.g. when you activate it. The processing of a function list can be configured to the event.

Runtime behavior

If the graphic IO field displays a cactus image, you have not defined a graphic to be output for a specific value in your project.

The contents of the graphic IO field change color to show that it is now activated.

The frame in 3D is only shown graphically in an output field.

Touch and key operation

Touch operation

Touch the graphic IO field on the touch screen of the HMI device. Selection mode is now enabled.

Select the graphic object using the scroll bar.

Touch the selected graphic object to save it or discard the selection by touching a different screen object.

Key operation

How to use a graphic IO field on the keyboard device:

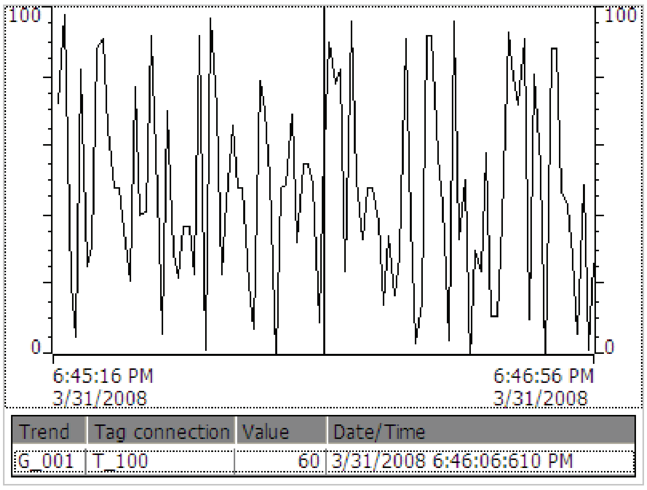
Step		Procedure	
1	Select the graphic IO field	e.g. ▶	The graphic IO field is marked.
2	Enable the selection mode	ENTER	The selection mode is now enabled.
3	Select an entry	▲ ▼ ◀ ▶	Moves the cursor by lines.
4	Accept selection or	ENTER	The entry selected is now valid. The selection mode is closed.
	Cancel selection	ESC	The original value is restored.

Trend view

Overview

Application

The trend view is a dynamic display object. The Trend view can display actual process data and process data from a log continuously when it is supported by the HMI device.



Trend	Tag connection	Value	Date/Time
G_001	T_100	60	3/31/2008 6:46:06:610 PM

Layout

The layout of the trend view is based on the configuration. A trend view can show multiple curves simultaneously to allow the user, for example, to compare different process sequences. If the displayed process value exceeds or falls below the configured limit values, the violation of the limit can be displayed by a change of color in the curve.

A ruler can also simplify the reading of the process values from the trend view. The ruler displays the Y-value that belongs to an X-value.

Operation

Depending on the configuration you can:

- The displayed time section extends.
- The displayed time section reduces.
- Scroll back by one display width.
- Scroll forwards by one display width.
- Stop or continue the trend recording.

Operator controls

The buttons have the following functions:

Operator control	Function
⏮	Scrolls back to the start of trend recording. The start values of the trend recording are displayed there.
⊕	Zooms the displayed time section
⊖	Zooms out of the display time section
◀┊	Moves the ruler backwards (to the left).
┊▶	Moves the ruler forward (to the right).
⏪	Scrolls back by one display width (to the left).
⏩	Scrolls forward by one display width (to the right).
┊	Shows or hides the ruler. The ruler displays the X-value associated with a Y-value.
■	Stops or continues trend recording

Touch and key operation

Procedure

Touch the relevant operator controls of the trend view on the touch screen of the HMI device.

Procedure

Select the trend view with the TAB key using the configured tab sequence.

The table below lists the key shortcuts available:

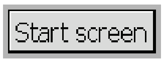
Keys	Function
CTRL ENTER	Scrolls back to the start of trend recording. The start values of the trend recording are displayed there.
CTRL + +	Enlarges the time section displayed.
CTRL + −	Reduces the time section displayed.
CTRL + ALT + ◀	Moves the ruler backwards (to the left).
CTRL + ALT + ▶	Moves the ruler forward (to the right).
SHIFT + ◀	Scrolls back by one display width (to the left).
SHIFT + ▶	Scrolls forward by one display width (to the right).

Button

Overview

Application

A button is a virtual key on the screen of the HMI device that can have one or more functions.



Layout

The layout of the button depends on the button type.

- Button with text: The text shown on the button gives information regarding the status of the button.
- Button with graphic: The graphic shown on the button gives information regarding the status of the button.
- Invisible: The button is not visible during Runtime.

Operation

Depending on the configuration, you can operate the button in the following ways:

- Standard operation: Click the button.
- Event: An event is triggered when you operate the button, e.g. when you click it. The processing of a function list can be configured to the event.

Runtime behavior

The operation may be followed with a optical feedback. However, note that the optical feedback only indicates a completed operation and not whether the configured functions were actually executed.

Touch and key operation

Procedure

Touch the button on the touch screen of the HMI device.

Procedure

How to operate a button on the keyboard device:

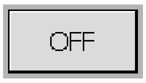
1. Select the button using a cursor key, e.g. ▶
2. Next, press the ENTER or the ␣ key.

Switch

Overview

Application

The switch is an operating element and display object with two states: "pressed" and "released." Switches can signal the state of a system component that cannot be seen from the HMI device, e.g. a motor. You can also change the state of that system component at the HMI device.



Layout

The layout of the switch depends on the switch type.

- Switches: The switch has a slider. The position of this slider gives information about the status of the switch.
- Switch with text: The text shown on the switch gives information regarding the status of the switch.
- Switch with graphic: The graphic shown on the switch gives information regarding the status of the switch.

Operation

Depending on the configuration, you can operate the switch in the following ways:

- Standard operation: Click the switch.
- Event: An event is triggered when you operate the switch, e.g. when you click it. The processing of a function list can be configured to the event.

Runtime behavior

A switch has two stable states: When you activate the switch, it changes to the other state. The switch retains this state until the next operation.

Touch and key operation

Touch operation

Touch operation of the switch differs, depending on its type:

- If a slider is displayed for the switch:

  Drag the slider to the new position on the touch screen of the HMI device or double-click in the slider range.
- If only a text or graphic object is displayed for the switch:

  Touch the switch on the touch screen of the HMI device.

Key operation

How to operate a switch on the keyboard device:

- Select the switch using a cursor key, e.g. ▶
- Next, press the ENTER or the ␣ key.

Symbolic I/O field

Overview

Application

A symbolic IO field can be assigned the following functions in Runtime:

- Output of text list entries
- Combined input and output

Example of its use as combination IO field:

You control a motor in Runtime by selecting the "Motor OFF" or "Motor ON" text from the text list. The motor is started or stopped in accordance with the selection. The symbolic IO field visualizes the current status of the motor.

Operation

The following options of operating the symbolic IO field are available, depending on the configuration:

- Standard operation: Select an entry from the text list.
- Event: You trigger an event by operating the symbolic IO field, e.g. by enabling it. The processing of a function list can be configured to the event.

Runtime behavior

The selection list of the symbolic IO field displays an empty text line if a corresponding entry was not defined in project data. The active state is indicated on the HMI device by changing the color of contents of the symbolic IO field.

Touch and key operation

Touch operation

Touch the symbolic IO field on the touch screen of the HMI device. The selection list displays the default entries.

If the selection list displays a scroll bar: Touch the scroll bar on the touch screen of the HMI device. Touch the scroll bar and drag to the required position on the touch screen.

Select the entry and accept the corresponding tag value by touching the entry on the screen. The selection list closes and the entry is displayed. The focus is still set on the symbolic IO field.

Key operation

How to operate a symbolic IO field on the keyboard device:

Step		Procedure	
1	Select the symbolic IO field	e.g. ▶	The symbolic IO field is marked.
2	Open selection list	ENTER	The drop down list box opens.
3	Select an entry	▲ ◀ ▼ ▶	Moves the cursor by lines.
4	Accept the selection Or	ENTER	The entry selected is now valid. The selection list is closed.
	Canceling the selection	ESC	The original value is restored. The selection list is closed.

10.11.4.3 Project security

Overview

Design of the security system

The configuration engineer can protect the operation of a project by implementing a security system.

The security system is based on authorizations, user groups and users.

If you use an operator control with access protection, the HMI device first requests that you log on. A logon screen is displayed in which you enter your user name and password. After logging on, you can operate the operator controls for which you have the necessary authorizations.

The logon dialog can be set up by the configuration engineer via an individual operator control.

In the same way, the configuration engineer can set up an operator control to log off. After logging off, objects assigned access protection can no longer be operated; to do so, log on again.

User groups and authorizations

Project-specific user groups are created by the configuration engineer. The "Administrators" and "Users" groups are included in all projects by default. User groups are assigned authorizations. Authorization required for an operation is specifically defined for each individual object and function in the project.

Users and passwords

Each user is assigned to exactly one user group.

The following people are allowed to create users and assign them passwords:

- The configuration engineer during configuration
- The administrator on the HMI device
- A user with user administration authorization on the HMI device

Irrespective of the user group, each user is allowed to change his own password.

Logoff times

A logoff time is specified in the system for each user. If the time between any two user actions, such as entering a value or changing screens, exceeds this logoff time, the user is automatically logged off. The user must then log on again to continue to operate objects assigned access protection.

Backup and restore

The user data is encrypted and saved on the HMI device to protect it from loss due to power failure.

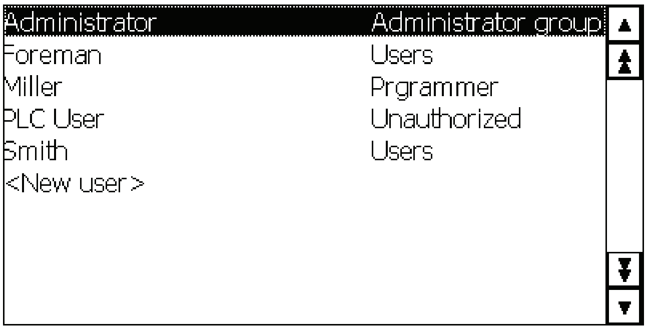
The users, passwords, group assignments and logoff times set up on the HMI device can be backed up and restored. This prevents you having to enter all of the data again on another HMI device.

NOTICE
The currently valid user data is overwritten in the following cases: • Depending on settings for a new download of the project. • Upon restore of a backed-up project • Upon import of the user administration via an operator control. The newly downloaded or restored user data and passwords take effect immediately.

Simple user view

Use

On HMI devices with a small display, the simple user view is used to display users on the HMI device.



Administrator	Administrator group
Foreman	Users
Miller	Prgrammer
PLC User	Unauthorized
Smith	Users
<New user>	

Note

The "Simple user view" object cannot be operated dynamically with a script.

Layout

The appearance depends on the authorizations.

- All users on the HMI device are displayed in the User view to the administrator or to a user with administrator authorizations.
- When user administration authorization is lacking, only the personal user entry is displayed.

Operation

Depending on the configuration you can:

- Manage users, e.g. create, delete.
- Change existing user data.
- Export or import user data.

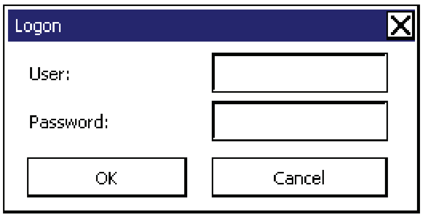
Note

On an HMI device the number is limited to 100 users and one PLC user. This restriction does not apply to PCs. On a PC, the maximum number of users is restricted by the physical memory.

User logon

Logon dialog

Use the logon dialog to log on to the security system of the HMI device. Enter your user name and password in the logon dialog.



The logon dialog opens in the following cases:

- You use an operator control with access protection.
- You press an operator control that was configured for displaying the logon dialog.
- You enable the "<ENTER>" entry in the simple user view.
- You enable a blank entry in the extended user view.
- The logon dialog will be automatically displayed when the project is started, depending on the configuration.

Requirement

The logon dialog is open.

Procedure for touch operation

Proceed as follows:

1. Enter the user name and password.

   Touch the corresponding input field. The alphanumerical screen keyboard is displayed.
2. Select "OK" to confirm logon.

Procedure for key operation

Proceed as follows:

1. Select the "User" input field within the logon dialog by pressing the TAB key.
2. Enter the user name using the system keys.
   Switch the numerical keypad to alphabet mode using the A-Z key to enter letters.
3. Use the TAB key to select the "Password" input field.
4. Enter the password using the system keys.
5. Confirm your entries with "OK".

Note

The user name is not case sensitive.

The password is case sensitive.

Result

After successful logon to the security system, you can execute functions on the HMI device which are access protected and for which you have authorization.

An alarm is output if an incorrect password has been entered and if an alarm view was configured.

User logoff

Requirement

You have logged into the security system of the HMI device.

Procedure

You have the following options for logging off:

- You press an operator control that was configured for logoff.
- You will be logged off automatically if you are not operating the project and if the logoff time has been exceeded.

You will also be automatically logged off if you enter an incorrect password.

Result

You are no longer logged into the project. In order to use an operator control with access protection, you first have to log on again.

Creating users

Requirement

- The user view is open.
- You are either authorized for user administration or you are an administrator.
- A user group has been created.

NOTICE

Runtime users must be assigned to a user group. The user group is created in the Engineering System. The designation of the user group is language-dependent.

NOTICE

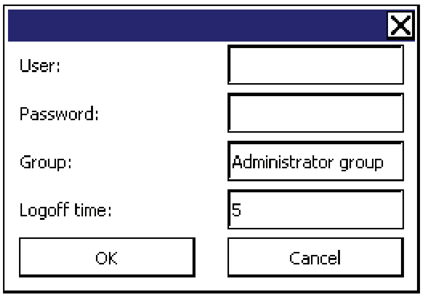
The following characters cannot be used in passwords:

- Blanks
- Special characters * ? . % / \ ' "

Creating users in the simple user view

Proceed as follows:

1. Click the "<New user>" entry in the user view. A dialog opens.



User:

Password:

Group: Administrator group

Logoff time: 5

2. Enter the desired user name and password.

   Touch the corresponding text box. The alphanumerical on-screen keyboard is displayed.
3. Click on the text box of the group. A dialog opens.
4. Assign the user to a group. Select ▲ and ▼ to scroll the selection list.
5. Touch the required entry in the drop down list box.

   The selected entry is accepted as input.
6. Touch the text box "Logoff time". The on-screen keyboard is displayed.
7. Enter a logoff time between 0 and 60 minutes. The value 0 stands for "no automatic logoff."
8. Confirm your entries with "OK."

Result

The new user is created.

Changing the user

Requirement

The user view is open.

Your authorization level determines the data you can change:

- You are an administrator or a user with user administration authorization. In these cases you are allowed to change the data for all the users on the HMI device in the user view:
  - User name
  - Group assignment
  - Password
  - Logoff time
- You are a user without user management authorization. In this case you are only allowed to change your personal user data:
  - Password
  - Logoff time, if configured

Note

You can only change the logoff time and password for the "Admin" user.

You can only change the logoff time for the "PLC_User". This user is used for logging on via the PLC.

NOTICE
Changes in the user view are effective immediately in Runtime. Changes in Runtime are not updated in the Engineering System. When the user management is downloaded to the HMI device, all changes in the user view are overwritten.

Changing user data in the simple user view

Proceed as follows:

1. In the user view, touch the user whose user data you want to change.
2. When entering the data, use exactly the same procedure as for creating a user.

Changing user data in the advanced user view

Proceed as follows:

1. In the user view, touch the user whose user data you want to change.
2. When entering the data, use exactly the same procedure as for creating a user.

Result

User data have been changed.

Deleting a user

Requirement

- You have opened a screen that contains the user view.
- You must be logged on with administrator rights or be authorized for user management to delete users.

> NOTICE
>
> Changes in the user view are effective immediately in Runtime. Changes in Runtime are not updated in the Engineering System.
>
> When the user management is downloaded to the HMI device, all changes in the user view are overwritten.

Procedure for touch operation

Proceed as follows:

1. Touch the user to be deleted in the user view.
2. Delete the user name.

Procedure for key operation

Proceed as follows:

1. Select the user view using the TAB key or the cursor keys.
2. Select the user in the user view by means of cursor keys.
3. Press INS DEL to delete the user.

Result

The user has been deleted and may no longer log onto the project.

10.11.4.4 Operating alarms

Overview

Alarms

Alarms indicate events and states on the HMI device which have occurred in the plant, in the process or on the HMI device itself. A status is reported when it is received.

An alarm could trigger one of the following alarm events:

- Incoming
- Outgoing
- Acknowledge

The configuration engineer defines which alarms must be acknowledged by the user.

An alarm may contain the following information:

- Date
- Time
- Alarm text
- Event text
- Location of fault
- Status
- Alarm class
- Alarm number
- Alarm group
- Supports diagnostics

Alarm classes

Alarms are assigned to various alarm classes. The selection depends on the HMI device.

- "Warnings"

  Alarms of this class usually indicate states of a plant such as "Motor switched on." Alarms in this class do not require acknowledgment.
- "Errors"

  Alarms in this class must always be acknowledged. Alarms normally indicate critical errors within the plant such as "Motor temperature too high".

- "System"

  System alarms indicate states or events which occur on the HMI device.

  System alarms provide information on occurrences such as operator errors or communication faults.
- "Diagnosis Events"

  SIMATIC diagnostic alarms show states and events in the SIMATIC S7 controllers.

Note

Availability for specific devices

Diagnostic alarms are not available for Basic Panels.

- STEP 7 alarm classes

  The alarm classes configured in STEP 7 are also available to the HMI device.

  Note

  Availability for specific devices

  STEP 7 alarm classes are not available for Basic Panels.
- Custom alarm classes

  The properties of this alarm class must be defined in the configuration.

Alarm buffer

Alarm events are saved to an internal buffer. The size of this alarm buffer depends on the HMI device type.

Alarm report

When alarm report is enabled in the project, alarm events are output directly to the connected printer.

You can set the reporting function separately for each alarm. The system outputs "Incoming" and "Outgoing" alarm events to the printer.

The output of alarms of the "System" alarm class to a printer must be initiated by means of the corresponding alarm buffer. This outputs the complete content of the alarm buffer to the printer. To be able to initiate this print function, you need to configure a corresponding control object in the project.

Note

Availability for specific devices

Alarm reports are not available for Basic Panels.

Alarm log

Alarm events are stored in an alarm log, provided this log file is configured. The capacity of the log file is limited by the storage medium and system limits.

Note

Availability for specific devices

Alarm logs are not available for Basic Panels.

Alarm view

The alarm view shows selected alarms or alarm events from the alarm buffer or alarm log. Whether alarms have to be acknowledged or not is specified in your configuration. By means of configuration, the display can be filtered in such a way that only alarms that contain a specific character string will be shown.

Alarm window

If configured, an alarm window shows all pending alarms or alarms awaiting acknowledgment of a particular alarm class. The alarm window is displayed as soon as a new alarm occurs.

You can configure the order in which the alarms are displayed. Either the current alarm or the oldest alarm is displayed. The alarm window can also be set to indicate the exact location of the fault, including the date and time of the alarm event. By means of configuration, the display can be filtered in such a way that only alarms that contain a specific character string will be shown.

Alarm indicator

The alarm indicator is a graphic icon that is displayed on the screen when an alarm of the specified alarm class is incoming.

The alarm indicator can assume one of two states:

- Flashing: At least one unacknowledged alarm is pending.
- Static: The alarms are acknowledged but at least one of them is not yet deactivated. The number displayed indicates the number of pending alarms.

Simple alarm view, alarm window

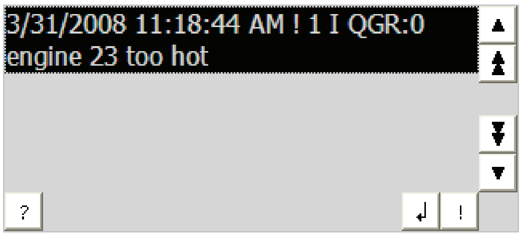
Use

The simple alarm view shows selected alarms or alarm events from the alarm buffer or alarm log. The layout and operation of the simple alarm window correspond to that of the simple alarm view.

Note

The "Single alarm view" object cannot be operated dynamically with a script.

In the Engineering System you can dynamically control the visibility of an object, for example, in the "Animations" tab of the Properties window. In Runtime, the "Simple alarm view" does not support animations. If you have configured an animation and wish to perform a consistency check of the project, for example, then an error alarm is issued in the Output window.



3/31/2008 11:18:44 AM ! 1 I QGR:0
engine 23 too hot

Layout

Depending on the configuration, in the alarm view different columns with information regarding an alarm or an alarm event are displayed.

To differentiate between the different alarm classes, the first column in the alarm view contains an icon:

Icon	Alarm class
!	"Errors"
(empty)	"Warnings"
(depends on the configuration)	Custom alarm classes
$	"System"

Operation

Depending on the configuration you can:

- Acknowledge alarms
- Edit alarms

Control elements

The buttons have the following functions:

Button	Function
!	Acknowledge alarm
↲	Edit alarm
?	Display infotext for an alarm
►	Shows the full text of the selected alarm in a separate window, the alarm text window. In the alarm text window, you can view alarm text that requires more space than is available in the alarm view. Close the alarm text window with ☒.
▲	Scrolls one alarm up
⇞	Scrolls one page up in the alarm view
⇟	Scrolls one page down in the alarm view
▼	Scrolls one alarm down

Layout of the operator controls

The layout of the buttons for operating the simple alarm view depends on the configured size. Therefore check if all the required buttons are available on the HMI device.

Detecting pending alarms

Introduction

You can recognize the presence of alarms which must be acknowledged by the following:

- For an HMI device with keys: The LED of the key ACK is lit.
- Depending on the configuration: An alarm indicator is displayed on screen.

The configuration determines whether an alarm has to be acknowledged or not. This is also defined by the alarm class which an alarm belongs to.

LED in the "ACK" key

An LED is integrated in the ACK key on HMI devices with keyboard. The LED is lit if there are alarms requiring acknowledgment which must still be acknowledged.

The LED goes out when you acknowledge all alarms requiring acknowledgment.

Alarm indicator

The alarm indicator is a graphic symbol indicating pending alarms or alarms requiring acknowledgment, depending on the configuration.

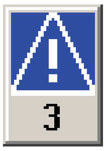


Figure 10-2 Alarm indicator with three pending alarms

Layout

The alarm indicator can assume one of two states:

- Flashing:

  The alarm indicator flashes as long as alarms are pending for acknowledgment. The number displayed indicates the number of pending alarms. The project engineer can configure specific functions to be executed by operating the alarm indicator.
- Static: The alarms are acknowledged but at least one of them is not yet deactivated.

Displaying dialogs

The displayed alarm indicator view is covered, for example, by the logon dialog, help dialog or alarm text windows. The alarm indicator is visible once you close these dialogs.

Display infotext for alarm

Procedure for touch operation

Proceed as follows to display the info text:

1. Touch the relevant alarm in the alarm view or in the alarm window.

   The alarm is selected.
2. Touch the ? button in the simple alarm view or ? in the advanced alarm view.

   If configured, the info text assigned to this alarm is displayed.
3. Close the screen for displaying the Info text by means of the ☒ button.

Procedure for key operation

Proceed as follows to display the info text:

1. Select the desired alarm in the alarm view.
2. Press the key HELP.

   If configured, the info text assigned to this alarm is displayed.
3. Close the info text by pressing the HELP key.

Acknowledge alarm

Requirement

The alarm to be acknowledged is displayed in the alarm window or the alarm view.

Procedure for touch operation

To acknowledge an alarm, proceed as follows:

1. Touch the relevant alarm in the alarm view or in the alarm window.

   The alarm is selected.
2. Touch the ! button in the simple alarm view or ✓ in the advanced alarm view.

Procedure for key operation

The alarm view and the alarm window have a tab sequence with which you can select operator controls and the last selected alarm using the keyboard.

To acknowledge an alarm, proceed as follows:

1. Select the desired alarm view or alarm window using the TAB key.
2. Select the desired alarm. Use the HOME, END, ▲ or ▼ keys accordingly.
3. Press the key ACK.

Alternative operation

Depending on the configuration, you can also acknowledge an alarm with a function key.

Result

The alarm is acknowledged. If the alarm belongs to an alarm group, all the alarms of the associated group are acknowledged.

Editing an alarm

Introduction

The configuration engineer can assign additional functions to each alarm. These functions are executed when the alarm is processed.

Note

When you edit an unacknowledged alarm, it is acknowledged automatically.

Requirement

The alarm to be edited is displayed in the alarm window or the alarm view.

Procedure for touch operation

Proceed as follows to edit an alarm:

1. Touch the relevant alarm in the alarm view or in the alarm window. The alarm is selected.
2. Touch the ↲ button in the simple alarm view or in the enhanced alarm view.

Procedure for key operation

Proceed as follows to edit an alarm:

1. Select the desired alarm view or alarm window with TAB.
2. Select the desired alarm. Use the HOME, END, ▲ or ▼ keys.
3. Continue to press the key TAB until the button ↲ is selected in the simple alarm view or in the extended alarm view.
4. Confirm your entry by pressing the key ENTER.

Result

The system executes the additional functions of the alarm. Additional information on this topic may be available in your plant documentation.

10.11.4.5 Operating recipes

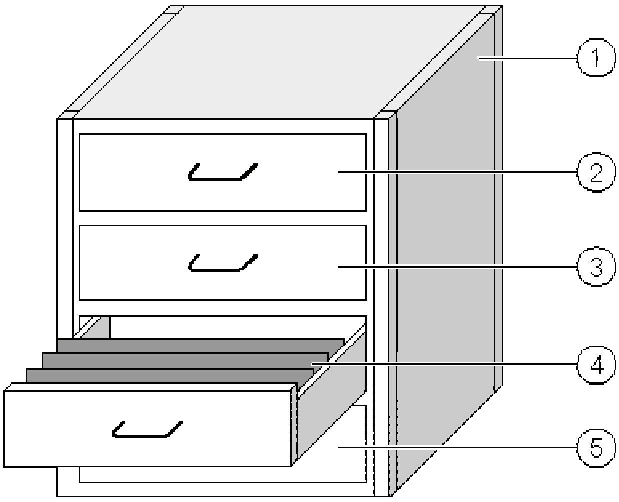
Structure of a recipe

Recipes

The recipe collection for the production of a product family can be compared to a file cabinet. A recipe which is used to manufacture a product corresponds to a drawer in a file cabinet.

Example:

In a plant for producing fruit juice, recipes are required for different flavors. There is a recipe, for example, for the flavors orange, grape, apple and cherry.



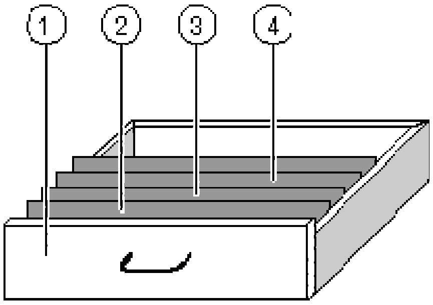
① File cabinet	Recipe collection	Recipes for a fruit juice plant
② Drawer	Recipe	Orange flavored drinks
③ Drawer	Recipe	Grape flavored drinks
④ Drawer	Recipe	Apple flavored drinks
⑤ Drawer	Recipe	Cherry flavored drinks

Recipe data records

The drawers of the file cabinet are filled with suspension folders. The suspension folders in the drawers represent records required for manufacturing various product variants.

Example:

Product variants of the flavor apple might be a soft drink, a juice or nectar, for example.



①	Drawer	Recipe	Product variants of apple flavored drinks
②	Suspension folder	Recipe data record	Apple drink
③	Suspension folder	Recipe data record	Apple nectar
④	Suspension folder	Recipe data record	Apple juice

Elements

In the figure showing the file cabinet, each suspension folder contains the same number of sheets. Each sheet in the suspension folder corresponds to an element of the recipe data record. All the records of a recipe contain the same elements. The records differ, however, in the value of the individual elements.

Example:

All drinks contain the same components: water, concentrate, sugar and flavoring. The records for soft drink, fruit juice or nectar differ, however, in the quantity of sugar used in production.

Recipes in the project

Overview

If recipes are used in a project, the following components interact:

- HMI device recipe memory

  Recipes are saved in the form of data records in the HMI device recipe memory.

- Recipe view / recipe screen

  On the HMI device, recipes are displayed and edited in the recipe view or in a recipe screen.

  - The recipe data records from the internal memory of the HMI device are displayed and edited in the recipe view.
  - The values of the recipe tags are displayed and edited in the recipe screen.

  Note

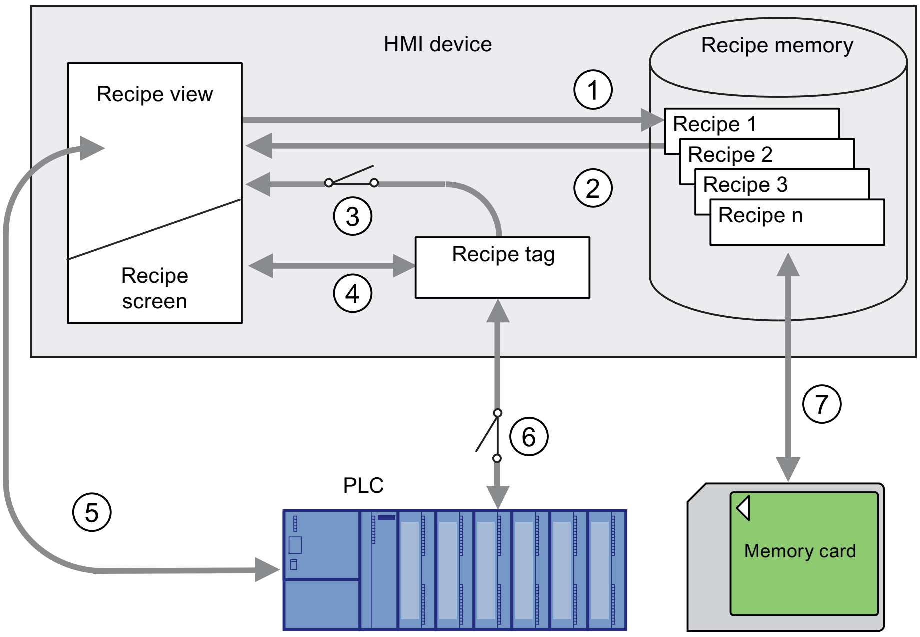
  The same recipe tags can be configured in a variety of recipes. If you modify the value of a recipe tag, the synchronization changes the value of the recipe tag in all recipes.

- Recipe tags

  The recipe tags contain recipe data. When you edit recipes in a recipe screen, the recipe values are stored in recipe tags. The point at which the recipe tag values are exchanged with the PLC depends on the configuration.

Data flow

The following figure shows the data flow in a project with recipes:



① Editing, saving or deleting a recipe data record

② Display recipe data record

③ Synchronize or do not synchronize recipe tags

④ Display and edit recipe tags in the recipe screen

⑤ Write records from the recipe view to the PLC or read records from the PLC and display them in the recipe view.

⑥ Recipe tags are to the PLC online or offline

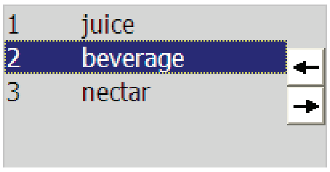
⑦ Export or import recipe data record to memory card

Simple recipe view

Layout

The simple recipe view consists of three areas:

- Recipe list
- Data record list
- Element list



1 juice
2 beverage
3 nectar

Figure 10-3 Simple recipe view - example with data record list

In the simple recipe view, each area is shown separately on the HMI device. You can use the shortcut menu to operate each of these display areas.

The simple recipe view always begins with the recipe list.

Operation

You can use the simple recipe view as follows, depending on the configuration:

- Create, change, copy or delete recipe data records
- Read recipe data records from the PLC or transfer to the PLC

Operator controls of the simple recipe view

Toggle between the display areas and the shortcut menus to operate the simple recipe views.

The table below shows the operation of the display area.

	Key	Function
Touching an entry	ENTER	The next lowest display area is opened, i.e. the data record list or the element list.
←	ESC	The previous display area opens.
→	▶	The shortcut menu of the display area opens.

The table below shows the operation of the shortcut menu:

	Key	Function
→	ESC	The menu is closed. The display area opens.
Touching the menu command	Input of the number of the menu command	The menu command is executed.

Shortcut menus of the simple recipe view

A shortcut menu can be called for each view area by pressing the → button or the ▶ key. The shortcut menu lists the commands that are available in the active view area. A number is assigned to each command. You execute the command by entering its number. You can also use the system keys to execute certain commands.

The scope depends on the HMI device.

- Recipe list

No.	Menu command	Keys	Function
0	New	SHIFT + INS DEL	A new recipe data record is created for the selected recipe. If a start value is configured, it is displayed in the input field.
1	Displaying infotext	HELP	Displays the infotext configured for the simple recipe view.
2	Open	ENTER	The record list of the selected recipe opens.

- Data record list

	Menu command	Keys	Function
	New	SHIFT + INS DEL	Creates a new recipe data record. If a start value is configured, it is displayed in the input field.
	Deleting	INS DEL	The displayed record is deleted.

	Menu command	Keys	Function
	Save as		The selected data record is saved under a different name. A dialog box opens where you can enter the name.
	Rename		Renames the selected data record. A dialog box opens where you can enter the name.
	Open	ENTER	The element list of the selected data record opens.
	Previous	ESC	The recipe list opens.
	To PLC		The displayed values of the selected data record are transferred from the HMI device to the PLC.
	From PLC		The recipe values from the PLC are displayed in the recipe view of the HMI device.
	Displaying infotext	HELP	Displays the infotext configured for the simple recipe view.

- Element list

	Menu command	Keys	Function
	Save		The selected record is renamed.
	To PLC		The displayed values of the selected data record are transferred from the HMI device to the PLC.
	From PLC		The recipe values from the PLC are displayed in the recipe view of the HMI device.
	Save as		The data record is saved under a new name. A dialog box opens where you can enter the name.
	Displaying infotext	HELP	Displays the infotext configured for the simple recipe view.
	Rename		Renames the selected data record. A dialog box opens where you can enter the name.
	Previous	ESC	The data record list opens.

Mouse control or touchscreen control of the simple recipe view

1. Select the desired recipe from the recipe view.
2. Click on the → button.

   The shortcut menu is opened.
3. Select the desired menu command.

   The menu command is executed.
4. Alternatively, open the desired recipe in the recipe view.

   The data record list is displayed.
5. Open the desired data record. You could also use the → button to open the shortcut menu and select a menu command.

   The menu command is executed.

Using the keyboard with the simple recipe view

1. Press the TAB key as many times as required to select the simple recipe view.
2. Select the desired recipe with the cursor keys.
3. Press the ▶ key.

   The shortcut menu is opened.
4. Press the ▼ cursor key as many times as required to select the menu command.
5. Confirm the menu command by pressing the key ENTER.
6. Alternatively, press the number of the desired menu command.

   The menu command is executed.

Creating a recipe data record

Introduction

Create a new recipe data record in the recipe list or in the record list. Then enter the values for the new record in the element list and save the record.

Requirement

A screen with a simple recipe view is displayed.

Procedure

Proceed as follows to create a recipe data record:

1. If the recipe list contains several recipes: Select the recipe for which you want to create a new recipe data record.
2. Open the recipe list menu.
3. Select the "0 new" menu command.

   Creates a new record.

   The element list of the new record opens.
4. Enter values for the data record elements.

   The tags of the record can be assigned default values depending on the configuration.
5. Open the menu of the element list and select the "0 Save" menu command.
6. Enter a name for the new record.
7. Confirm your entries.

   An existing data record is overwritten if you assign its number to a new data record.

Result

The new recipe data record is saved to the selected recipe.

Editing a recipe data record

Introduction

Edit the values of the recipe data records and save them to a recipe view.

Synchronization with the PLC

To display the current recipe values from the PLC in the simple recipe view, you first have to read the actual values in the element list from the PLC using menu command "2 From PLC".

Values changed in the recipe view are only activated after you transferred the modified data record to the PLC by means of menu command "1 To PLC".

Requirement

A screen with a recipe view is displayed.

Procedure

Proceed as follows to copy a recipe data record:

1. If the recipe list contains several recipes: Select the recipe which contains the relevant recipe data record.
2. Open the data record list.
3. Select the recipe data record you want to edit.
4. Open the element list.
5. Edit the element values as required.
6. Save your changes using menu command "0 Save".

Result

The edited recipe data record is saved to the selected recipe.

Deleting a recipe data record

Introduction

You can delete all the data records which are not required.

Requirement

A screen with a simple recipe view is displayed.

Procedure for touch operation

Proceed as follows to delete a new recipe data record:

1. If the recipe list contains several recipes: Select the recipe which contains the relevant recipe data record.
2. Open the data record list.
3. Select the data record you want to delete.
4. Open the menu.
5. Select the menu command "1 Delete".

Procedure for key operation

Proceed as follows to delete a new recipe data record:

1. If the recipe list contains several recipes: Select the recipe which contains the relevant recipe data record.
2. Open the data record list.
3. Select the data record you want to delete.
4. Press INS DEL.

Result

The data record is deleted.

Reading a recipe data record from the PLC

Introduction

The values of recipe elements are exchanged with the PLC via tags.

You can edit values which were saved to the recipes in the HMI device directly at plant level within the active project; this may be the case if a valve in the plant opens more than is indicated in the recipe. The values of the tags on the HMI device possibly no longer match the values in the PLC.

Read the values from the PLC and output these to the recipe view to synchronize the recipe values.

Requirement

A screen with a simple recipe view is displayed.

Procedure

To transfer a recipe data record to the PLC, proceed as follows:

1. If the recipe list contains several recipes: Select the recipe which contains the relevant recipe data record.
2. Select the element list of the recipe data record to which you want to apply the values from the PLC.
3. Open the menu.
4. Select menu command "2 From PLC".

   The values are read from the PLC.
5. Save the displayed values to the HMI device using menu command "0 Save".

Result

The values were read from the PLC, are visible on the HMI device and were saved to the selected recipe data record.

Note

Basic Panels

With Basic Panels, the "From PLC" menu command can also be configured for the data record list: In this case, you can also select the "From PLC" menu command in the data record list.

Transferring a recipe data record to the PLC

Introduction

You must transfer the values to the PLC to activate a changed recipe data record for the process.

The values displayed in the recipe view are transferred to the PLC.

Requirement

A screen with a simple recipe view is displayed.

Procedure

To transfer a recipe data record to the PLC, proceed as follows:

1. If the recipe list contains several recipes: Select the recipe which contains the relevant recipe data record.
2. Select the element list of the recipe data record whose values you want to transfer to the PLC.
3. Open the menu.
4. Select menu command "1 To PLC".

Result

The values of the recipe data record were transferred to the PLC and are active in the process.

Note

Basic Panels

With Basic Panels, the "To PLC" menu command can also be configured for the data record list: In this case, you can also select the "To PLC" menu command in the data record list.

10.12 Performance features

10.12.1 Engineering system

Engineering system

The following tables help you assess whether your project still meets the performance specifications of the Engineering System.

In addition to the specified limits, allowances must also be made for restrictions imposed by main memory resources. WinCC uses up to 2 GB of RAM, depending on the operating system. It is nonetheless useful to install more than 2 GB of main memory on the PC if running many applications with high memory requirements in parallel.

Project system limits

	WinCC
Number of HMI devices in the project	35
Number of HMI tags [1)]	80.000
Number of logging tags	8.000
Number of blocks (faceplates, user data types)[3)]	10.000
Number of screens	3.000
Number of screen objects per screen	3.000
Number of screen objects	320.000
Number of alarms [2)] [3)]	20.000
Number of texts [3)]	300.000
Number of text lists and graphic lists [3)]	10.000
Number of entries per text list	3.000
Number of languages	32
Number of global libraries [3)]	20
Number of objects in the project library [3)]	300.000

1) Including logging tags.

2) With an average of 5 texts and a dynamic parameter

3) Including the objects configured in the "Program PLC" area

HMI device system limits

	WinCC
Number of HMI tags [1]	80.000
Number of logging tags	8.000
Number of logs	500
Number of screens	1000
Number of screen objects per screen	3.000
Number of screen objects	320.000
Number of function lists	30.000
Number of animations and local scripts	50.000
Number of user-defined functions	1.000
Number of tasks	500
Number of alarms [2]	20.000
Number of recipes	1.000
Number of recipe elements	10.000
Number of texts	100.000
Number of text lists and graphic lists	1.000
Number of entries per text list	3.000
Number of users	200
Number of reports	300

1) Including logging tags.

2) With an average of 5 texts and a dynamic parameter

System limits during migration

You can migrate projects which are beyond the specified system limits in one or more areas.

An alarm will be output if migration creates a project whose limits are beyond the specified system limits. You must then adapt the project after migration to within the specified system limits to ensure safe operation in WinCC.

10.12.2 Basic Panel

Basic Panel

The following table helps you assess whether your project meets the performance features of the HMI device.

The specified maximum values are not additive. It cannot be guaranteed that configurations running on the devices at the full system limits will be functional.

In addition to the specified limits, allowances must be made for restrictions imposed by configuration memory resources.

Tags

	KP300 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of tags in the project	250	250	500	500	500
Number of PowerTags	--	--	--	--	--
Number of elements per array	100	100	100	100	100
Number of local tags	--	--	--	--	--

Alarms

	KP300 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of alarm classes	32	32	32	32	32
Number of discrete alarms	200	200	200	200	200
Number of analog alarms	15	15	15	15	15
Length of an alarm in characters	80	80	80	80	80
Number of process values per alarm	8	8	8	8	8
Size of the alarm buffer	256	256	256	256	256
Number of queued alarm events	64	64	64	64	64

Screens

	KP300 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of screens	50	50	50	50	50
Number of fields per screen	30	30	30	30	30
Number of tags per screen	30	30	30	30	30
Number of complex objects per screen 1)	5	5	5	5	5

1) Complex objects include: bars, sliders, symbol library, clock, and all objects from the Controls area.

Recipes

	KP300 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of recipes	5	5	5	5	5
Number of elements per recipe1)	20	20	20	20	20
User data length in bytes per data record	--	--	--	--	--
Number of data records per recipe	20	20	20	20	20
Reserved memory for data records in the internal Flash	40 KB	40 KB	40 KB	40 KB	40 KB

1) If arrays are used, each array element counts as one recipe element

Logs

	KP300 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of logs	--	--	--	--	--
Number of entries per log (including all log segments) 1)	--	--	--	--	--
Number of log segments	--	--	--	--	--
Cyclic trigger for tag logging	--	--	--	--	--
Number of tags that can belogged per log	--	--	--	--	--

1) The number of entries for the "segmented circular log" logging method is the maximum number for all segmental circular logs. The product of the number of segmental circular logs and the number of data records per segmental circular log may not exceed the system limit.

Trends

	KP300 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of trends	25	25	25	25	25

Text lists and graphics lists

	KP300 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of graphics lists	100	100	100	100	100
Number of text lists	150	150	150	150	150
Number of entries per text or graphics list	30	30	30	30	30
Number of graphic objects	500	500	500	500	500
Number of text elements	500	500	500	500	500

Scripts

	KP300 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of scripts	--	--	--	--	--

Communication

	KP300 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of connections	4	4	4	4	4
Number of connections based on "SIMATIC HMI HTTP"	--	--	--	--	--

Help system

	KP300 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of characters in a help text	320	320	320	320	320

Languages

	KP300 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of runtime languages	5	5	5	5	5

Scheduler

	KP300 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Time-triggered tasks 1)	--	--	--	--	--

1) Event-triggered tasks are irrelevant for the system limits

User administration

	KP300 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of user groups	50	50	50	50	50
Number of authorizations	32	32	32	32	32
Number of users	50	50	50	50	50

Project

	KP300 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Size of the project file "*.srt"	512 KB	512 KB	512 KB	1024 KB	1024 KB

10.12.3 General technical data

10.12.3.1 Recommended printers

Recommended printers

The current list of printers recommended for use with the HMI devices is available on the Internet at:

Link to the current printer list

Note

All HMI devices except for a PC and Panel PC support only one printer at their USB port, even if several ports are available.

10.12.3.2 Memory requirements of recipes for Basic Panels

Introduction

The following calculation of memory requirements of recipes is valid for Basic Panels, OP 77A, and TP 177A devices.

Restrictions

The HMI device provides 39 KB of memory space for recipes. This memory space may not be exceeded. The total memory space for recipes is calculated as follows: Total of all recipes + recipe with highest memory requirement.

Each recipe may not exceed a maximum memory space of 19 KB.

Calculation of memory requirements

The memory space requirement of each recipe (in KB) is calculated based on the three addends D1 + D2 + D3.

Valid is::

- D1 = number of data records x M

  Rule for M (size of a data record):

  M = 1 x number of elements of a byte + 2 x number of elements of 2 bytes + 4 x number of elements of 4 bytes + 8 x number of elements of 8 bytes + K

  Rule for K (size of the string elements):

  K = number of string elements x (string length + 1) x 2

- D2 - data record size

  D2 = 4 + number of languages x 8 + number of languages x (4 + 4 x number of data records + (length of the data record name + 1) x 2 x number of data records) + 8 + 8 x number of data records

  Or rewritten:

  D2 = 12 + 8 x number of data records + number of languages x (12 + number of data records x (4 + (length of the data record name +1) x 2))

- D3 - shared memory

  D3 = 14 + number of elements

Note

Arrays and single elements can be calculated as described above.

10.13 Migration to WinCC V11

10.13.1 Overview of migration to WinCC V11

Overview of the section "Migration to WinCC V11"

SIMATIC WinCC V11 offers a number of functional changes. Some functions differ from the functions that you know from familiar environments such as WinCC V7 or WinCC flexible.

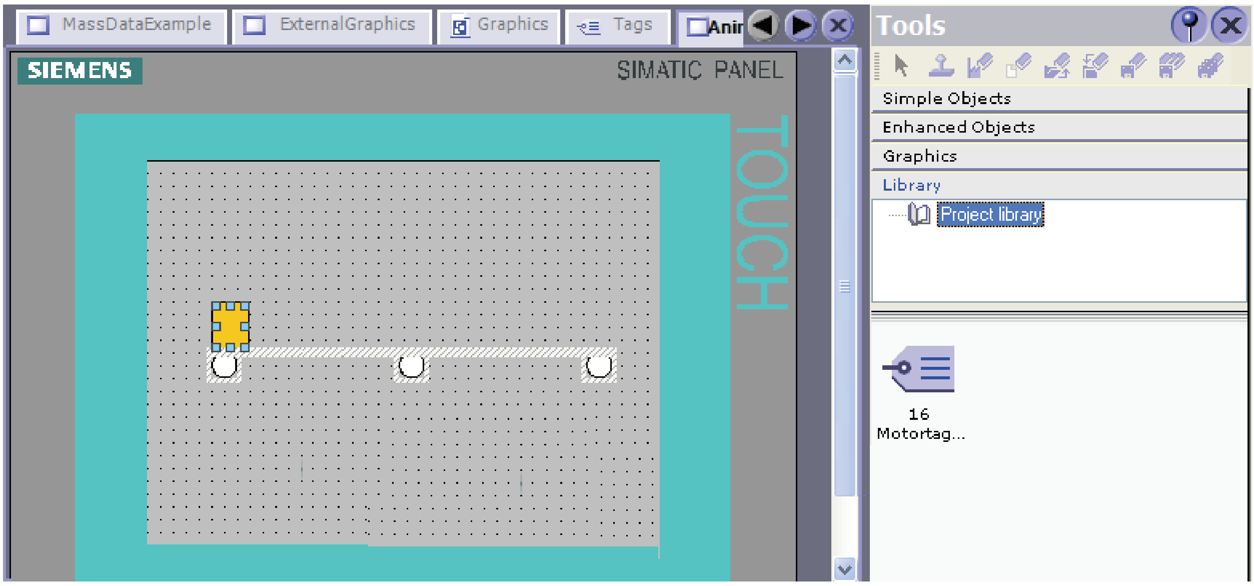
This document provides an overview of the special functions and procedures in SIMATIC WinCC V11.

These functions and procedures are fundamentally different from the WinCC V7 and WinCC flexible version, or have a different name.

10.13.2 Libraries

Libraries in WinCC flexible

Libraries are a collection of pre-configured screen objects. They expand the number of available screen objects and increase engineering efficiency, because library objects are always available for reuse; there is no need to reconfigure them.



WinCC flexible enables you to create two library types:

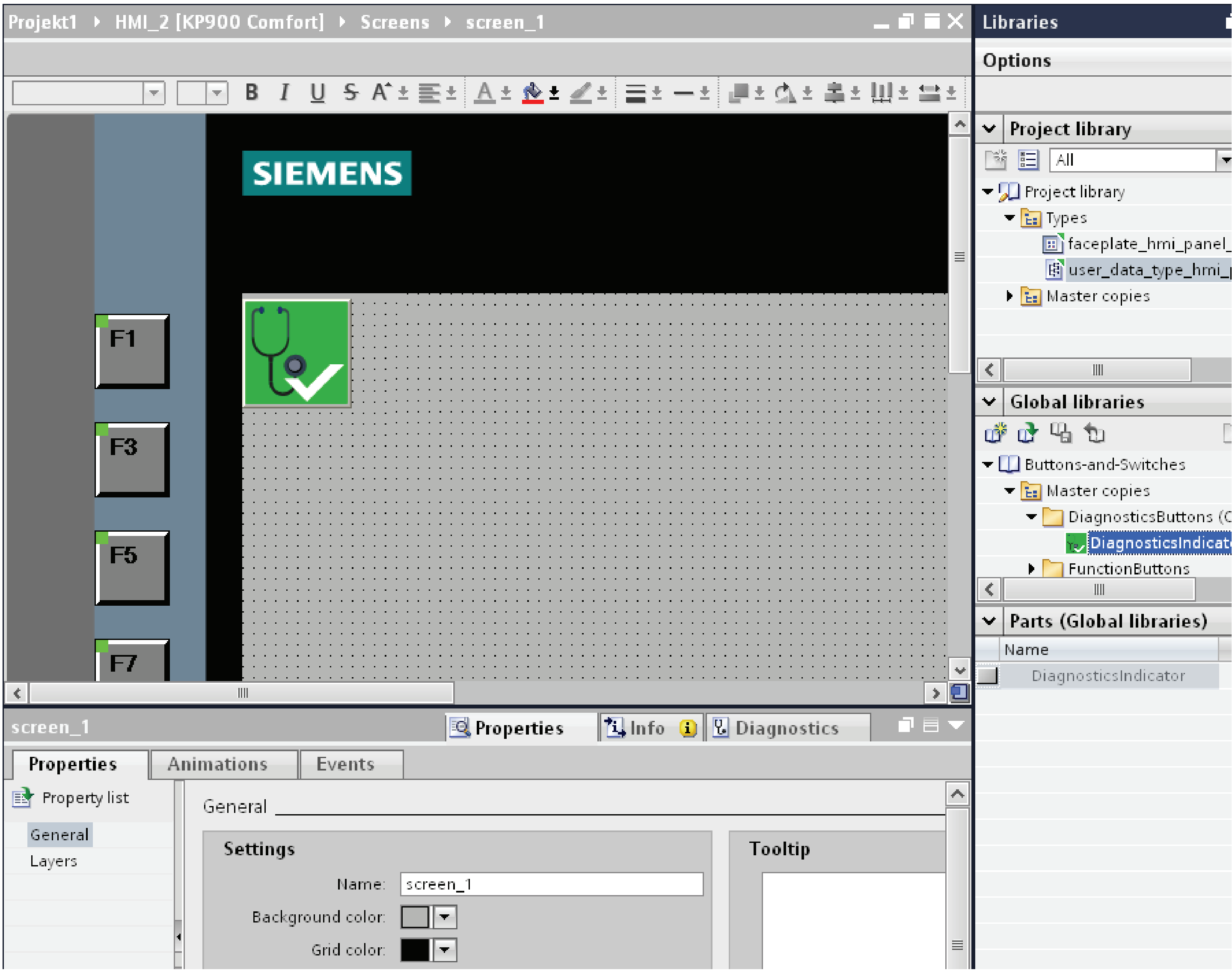
- Project library
- Global library

A library can contain all the WinCC flexible objects, such as screens, tags, graphic objects, or alarms.

How do I configure libraries in WinCC V11?

In WinCC V11, you also configure the "Project library" and the "Global library".

You can no longer store any system functions in libraries, as was the case in WinCC flexible.



Name: screen_1
Background color:
Grid color:

Both the "Project library" and the "Global library" contain the two folders, "Copy templates" and "Types". You can create or use the library objects as a copy template, or as a type.

- Copy templates

  Use copy templates to create independent copies of the library object.
- Types

  Create instances of objects of the "Types" folder and use the instances in your project. The instances are bound to their respective type. Changes to an instance also change all other instances. Types are marked by a green triangle in the "Libraries" task card.
- Managing the library objects

  You can only copy and move library objects within the same library.

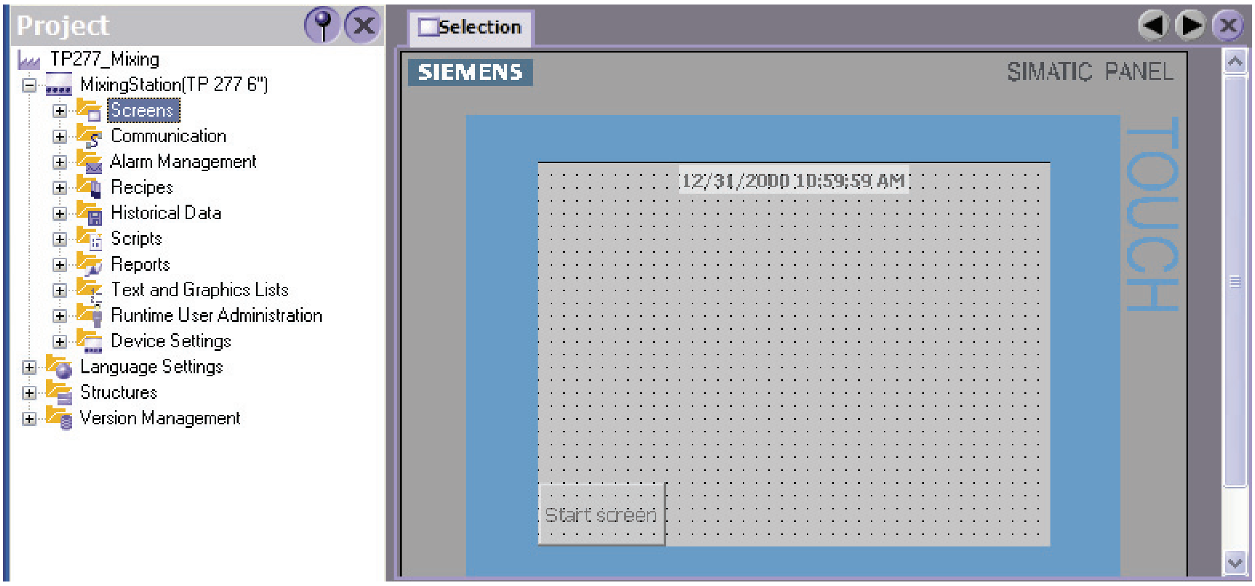
For more detailed information, see:

Libraries in WinCC (Page 2072)

10.13.3 Screens and templates

Screens and templates in WinCC flexible

In WinCC flexible, you create screens that an operator can use to control and monitor machines and plants. When you create your screens, the object templates provided support you in visualizing your plant, displaying processes and defining process values.



The project has a template for every HMI device. You can centrally configure the function keys and objects for your project in these templates.

Every screen based on this template will contain the function keys and objects that you configured in the template. Changes to an object or of a function key assignment in the template are applied to the object in all the screens, which are based on this template.

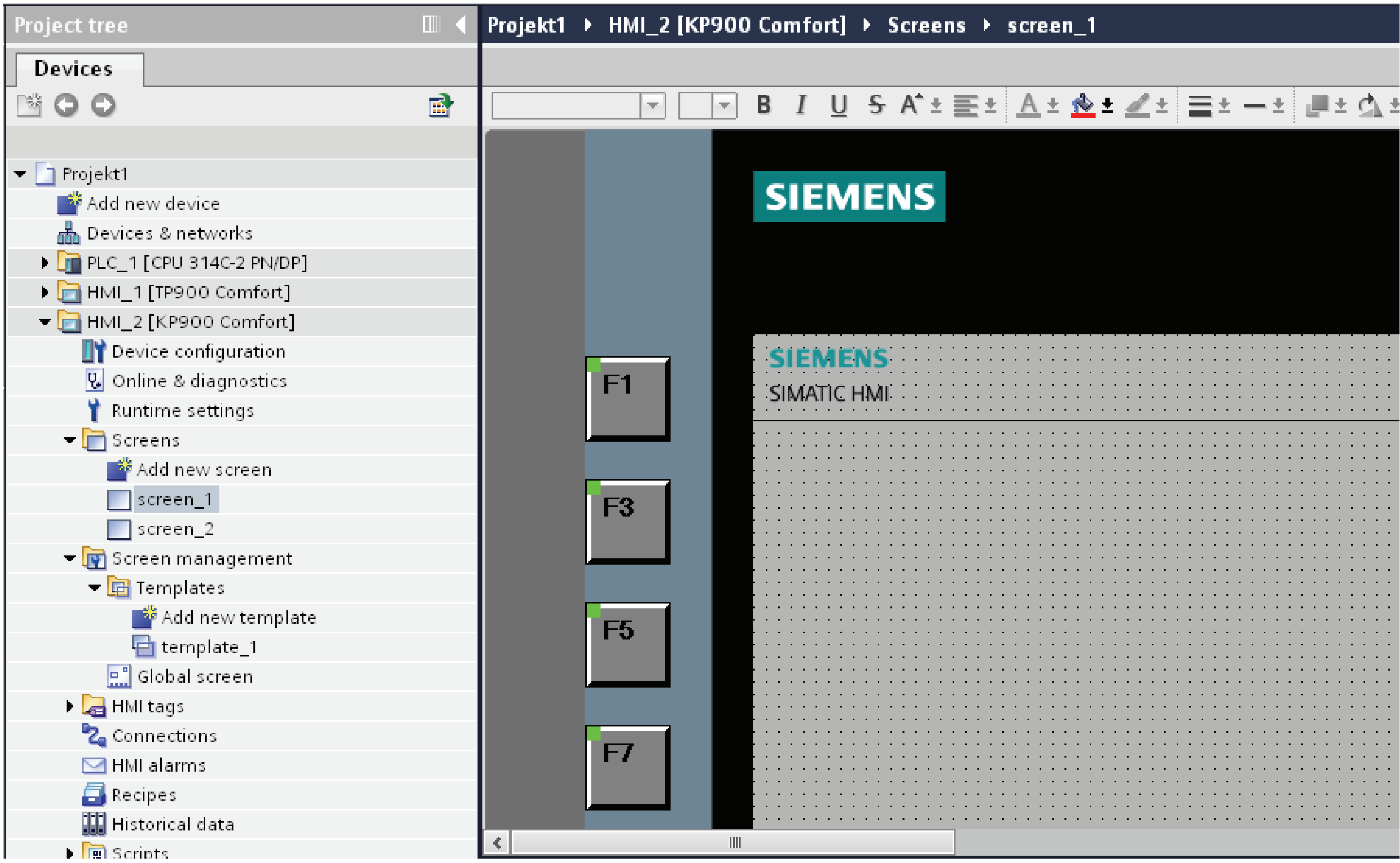
How do I configure screens and templates in WinCC V11?

In WinCC V11, you also configure "Templates" and a "Global screen" along with the "Screens".

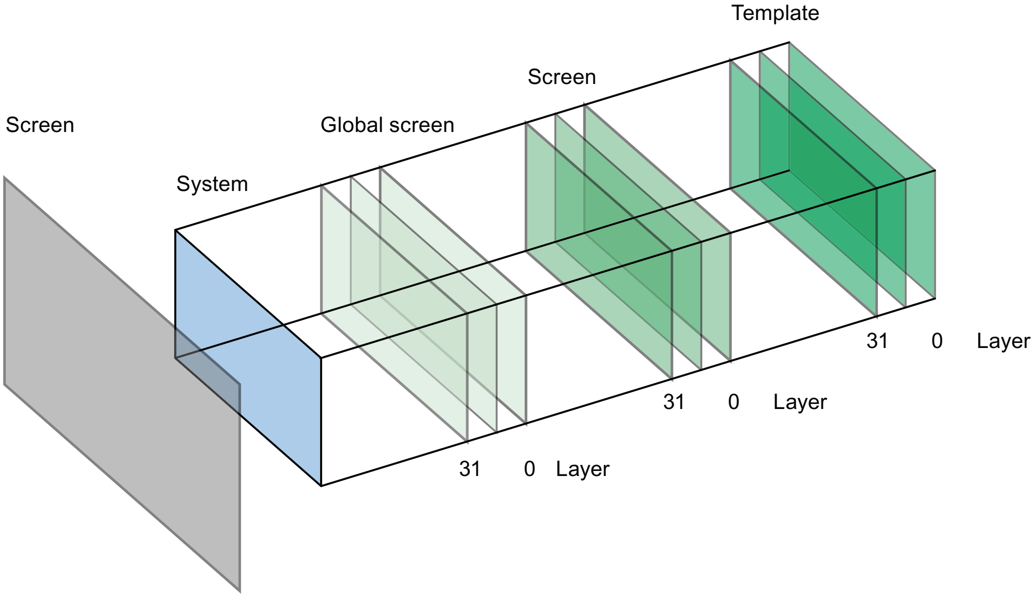
You determine functions and objects in the template which then apply to all screens based on this template. You can create multiple templates in WinCC.

In the "Global screen", define the elements which are independent of the template used for all screens of an HMI device. The "Alarm window" and "Alarm indicator" objects are available for use as global objects. For HMI devices with function keys, configure the function keys in the "Global Screen" editor.

For Comfort Panels, you also have the possibility of configuring a "System diagnostics view" in the global screen.



Excluding the controls, the screens are displayed in runtime in the following order:



For more detailed information, see:

Screen basics (Page 1967)

Using technology functions 11

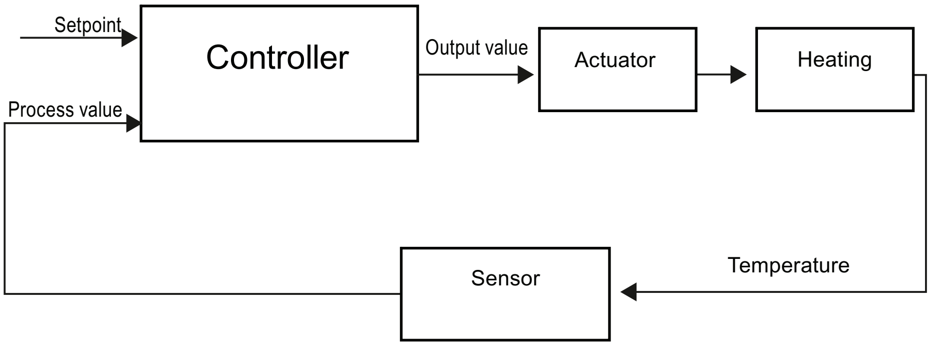
11.1 PID control

11.1.1 Principles for control

11.1.1.1 Controlled system and actuators

Controlled system

Room temperature control by means of a heating system is a simple example of a controlled system. A sensor measures the room temperature and transfers the value to a controller. The controller compares the current room temperature with a setpoint and calculates an output value (manipulated variable) for heating control.



A properly set PID controller reaches this setpoint as quickly as possible and then holds it a constant value. After a change in the output value, the process value often changes only with a time delay. The controller has to compensate for this response.

Actuators

The actuator is an element of the controlled system and is influenced by the controller. Its function modifies mass and energy flows.

The table below provides an overview of actuator applications.

Application	Actuator
Liquid and gaseous mass flow	Valve, shutter, gate valve
Solid mass flow, e.g., bulk material	Articulated baffle, conveyor, vibrator channel
Flow of electrical power	Switching contact, contactor, relay, thyristor
	Variable resistor, variable transformer, transistor

Actuators are distinguished as follows:

- Proportional actuators with constant actuating signal

  These elements set degrees of opening, angular positions or positions in proportion to the output value. The output value has an analog effect on the process within the control range.

  Actuators in this group include spring-loaded pneumatic drives, as well as motorized drives with position feedback for which a position control system is formed.

  An continuous controller, such as PID_Compact, generates the output value.

- Proportional actuators with pulse-width modulated signal

  These actuators are used to generate the output of pulses with a length proportional to the output value within the sampling time intervals. The actuator - e.g. a heating resistor or cooling apparatus - is switched on in isochronous mode for durations that differ depending on the output value.

  The actuating signal can assume unipolar "On" or "Off" states, or represent bipolar states such as "open/close", "forward/backward", "accelerate/brake".

  The output value is generated by a two-step controller such as PID_Compact with pulse-width modulation.

- Actuators with integral action and three-step actuating signal

  Actuators are frequently operated by motors with an on period that is proportional to the actuator travel of the choke element. This includes elements such as valves, shutters, and gate valves. In spite of their different design, all of these actuators follow the effect of an integral action at the input of the controlled system.

  A step controller, such as PID_3Step. generates the output value.

11.1.1.2 Controlled systems

The properties of a controlled system can hardly be influenced as these are determined by the technical requirements of the process and machinery. Acceptable control results can only be achieved by selecting a suitable controller type for the specific controlled system and adapting the controller to the time response of the controlled system. Therefore, it is is indispensable for the configuration of the proportional, integral and derivative actions of the controller to have precise knowledge of the type and parameters of the controlled system.

Controlled system types

Controlled systems are classified based on their time response to step changes of the output value.

We distinguish between the following controlled systems:

- Self-regulating controlled systems
  - Proportional-action controlled systems
  - PT1 controlled systems
  - PT2 controlled systems
- Non-self-regulating controlled systems
- Controlled systems with and without dead time

Self-regulating controlled systems

Proportional-action controlled systems

In proportional-action controlled systems, the process value follows the output value almost immediately. The ratio between the process value and output value is defined by the proportional Gain of the controlled system.

Examples:

- Gate valve in a piping system
- Voltage dividers
- Step-down function in hydraulic systems

PT1 controlled systems

In a PT1 controlled system, the process value initially changes in proportion to the change of the output value. The rate of change of the process value is reduced as a function of the time until the end value is reached, i.e., it is delayed.

Examples:

- Spring damping system
- Charge of RC elements
- Water container that is heated with steam.

The time constants are often identical for heating and cooling processes, or for charging and discharge characteristics. With different time constants, controlling is clearly more complex.

PT2 controlled systems

In a PT2 controlled system, the process value does not immediately follow a step change of the output value, i.e., it increases in proportion to the positive rate of rise and then approaches the setpoint at a decreasing rate of rise. The controlled system shows a proportional response characteristic with second order delay element.

Examples:

- Pressure control
- Flow rate control
- Temperature control

Non-self-regulating controlled systems

Non-self-regulating controlled systems have an integral response. The process value approaches an infinite maximum value.

Example:

- Liquid flow into a container

Controlled systems with dead time

A dead time always represents the runtime or transport time that has to expire before a change to the system input can be measured at the system output.

In controlled systems with dead time, the process value change is delayed by the amount of the dead time.

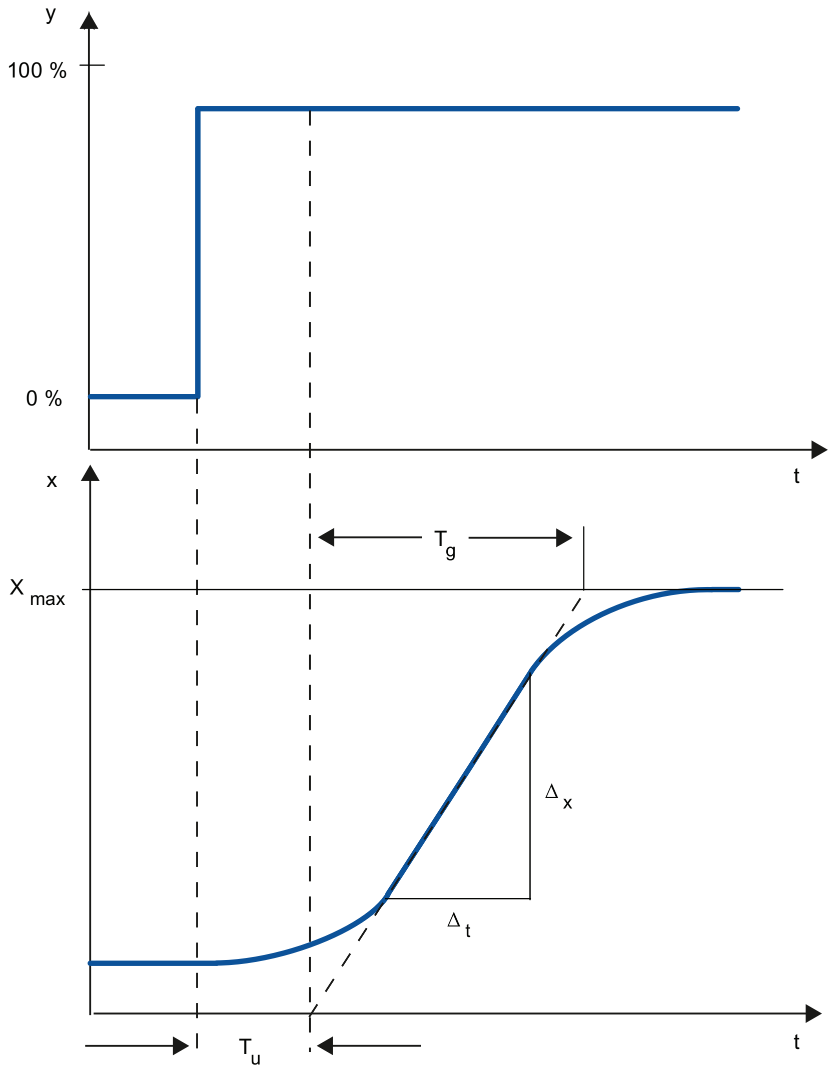
Example:

Conveyor

11.1.1.3 Characteristic values of the control section

Determining the time response from the step response

Time response of the controlled system can be determined based on the time characteristic of process value x following a step change of output value y. Most controlled systems are self-regulating controlled systems.



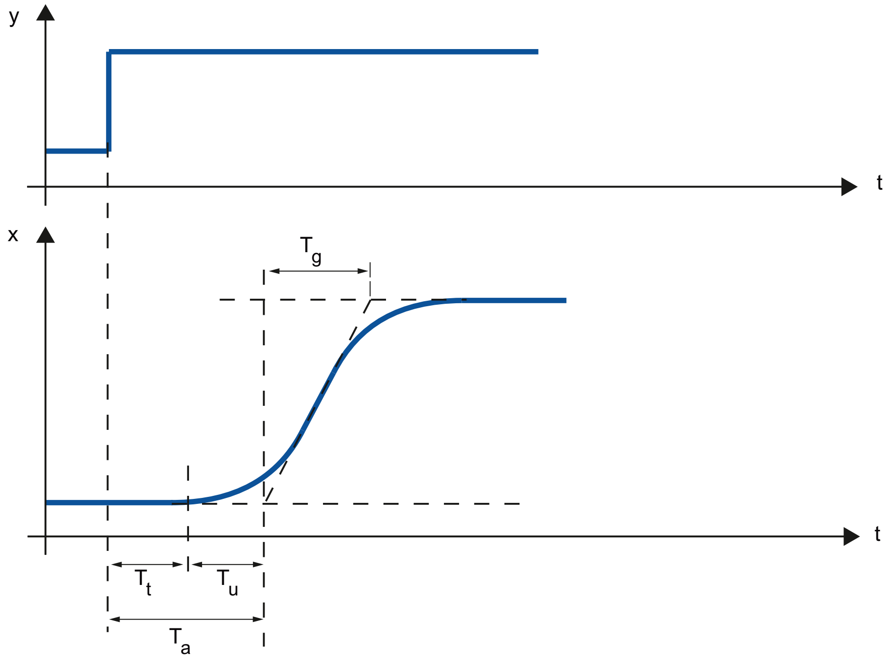
The time response can be determined by approximation using the variables Delay time T_u, Recovery time T_g and Maximum value X_{max}. The variables are determined by applying tangents to the maximum value and the inflection point of the step response. In many situations, it is not possible to record the response characteristic up to the maximum value because the process value cannot exceed specific values. In this case, the rate of rise v_{max} is used to identify the controlled system ($v_{max} = \Delta_x/\Delta_t$).

The controllability of the controlled system can be estimated based on the ratio T_u/T_g, or $T_u \times v_{max}/X_{max}$. Rule:

Process type	T_u / T_g	Suitability of the controlled system for controlling
I	< 0,1	can be controlled well
II	0.1 to 0.3	can still be controlled
III	> 0,3	difficult to control

Influence of the dead time on the controllability of a controlled system

A controlled system with dead time and recovery reacts as follows to a jump of the output value.



T_t	Dead time
T_u	Delay time
T_g	Recovery time
y	Output value
x	Process value

The controllability of a self-regulating controlled system with dead time is determined by the ratio of T_t to T_g. T_t must be small compared to T_g. Rule:

$T_t/T_g \leq 1$

Response rate of controlled systems

Controlled systems can be judged on the basis of the following values:

T_u < 0.5 min, T_g < 5 min = fast controlled system

T_u > 0.5 min, T_g > 5 min = slow controlled system

Parameters of certain controlled systems

Physical quantity	Controlled system	Delay time T_u	Recovery time T_g	Rate of rise v_{max}
Temperature	Small electrically heated furnace	0.5 to 1 min	5 to 15 min	Up to 60 K/min.
	Large electrically heated annealing furnace	1 to 5 min	10 to 20 min	Up to 20 K/min.
	Large gas-heated annealing furnace	0.2 to 5 min	3 to 60 min	1 to 30 K/min
	Distillation tower	1 to 7 min	40 to 60 min	0.1 to 0.5° C/s
	Autoclaves (2.5 m^3)	0.5 to 0.7 min	10 to 20 min	Not specified
	High-pressure autoclaves	12 to 15 min	200 to 300 min	Not specified
	Steam superheater	30 s to 2.5 min	1 to 4 min	2°C/s
	Injection molding machines	0.5 to 3 min	3 to 30 min	5 to 20 K/min
	Extruders	1 to 6 min	5 to 60 min	
	Packaging machines	0.5 to 4 min	3 to 40 min	2 to 35 K/min
	Room heating	1 to 5 min	10 to 60 min	1° C/min
Flow rate	Pipeline with gas	0 to 5 s	0.2 to 10 s	Not relevant
	Pipeline with liquid	None	None	
Pressure	Gas pipeline	None	0.1 s	Not relevant
	Drum boiler with gas or oil firing	None	150 s	Not relevant
	Drum boiler with impact grinding mills	1 to 2 min	2 to 5 min	Not relevant
Vessel level	Drum boiler	0.6 to 1 min	Not specified	0.1 to 0.3 cm/s
Speed	Small electric drive	None	0.2 to 10 s	Not relevant
	Large electric drive	None	5 to 40 s	Not relevant
	Steam turbine	None	Not specified	50 min^{-1}
Voltage	Small generators	None	1 to 5 s	Not relevant
	Large generators	None	5 to 10 s	Not relevant

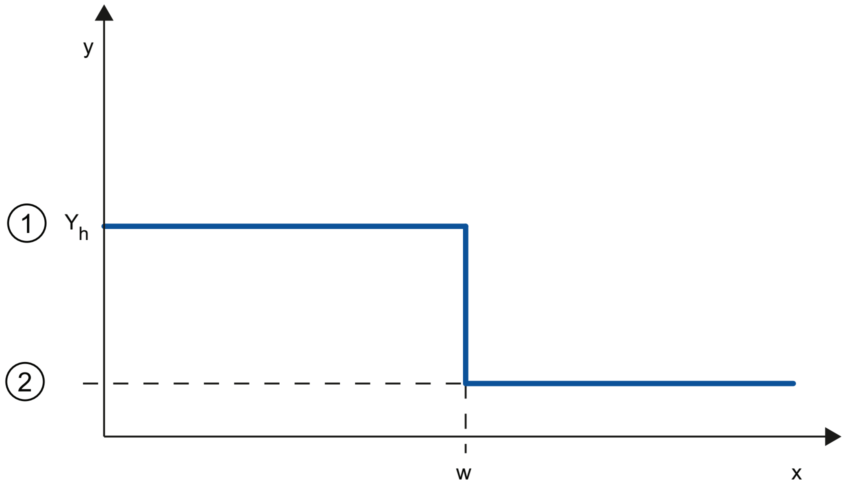
11.1.1.4 Pulse controller

Two-step controllers without feedback

Two-step controllers have the state "ON" and "OFF" as the switching function. This corresponds to 100% or 0% output. This behavior generates a sustained oscillation of process value x around setpoint w.

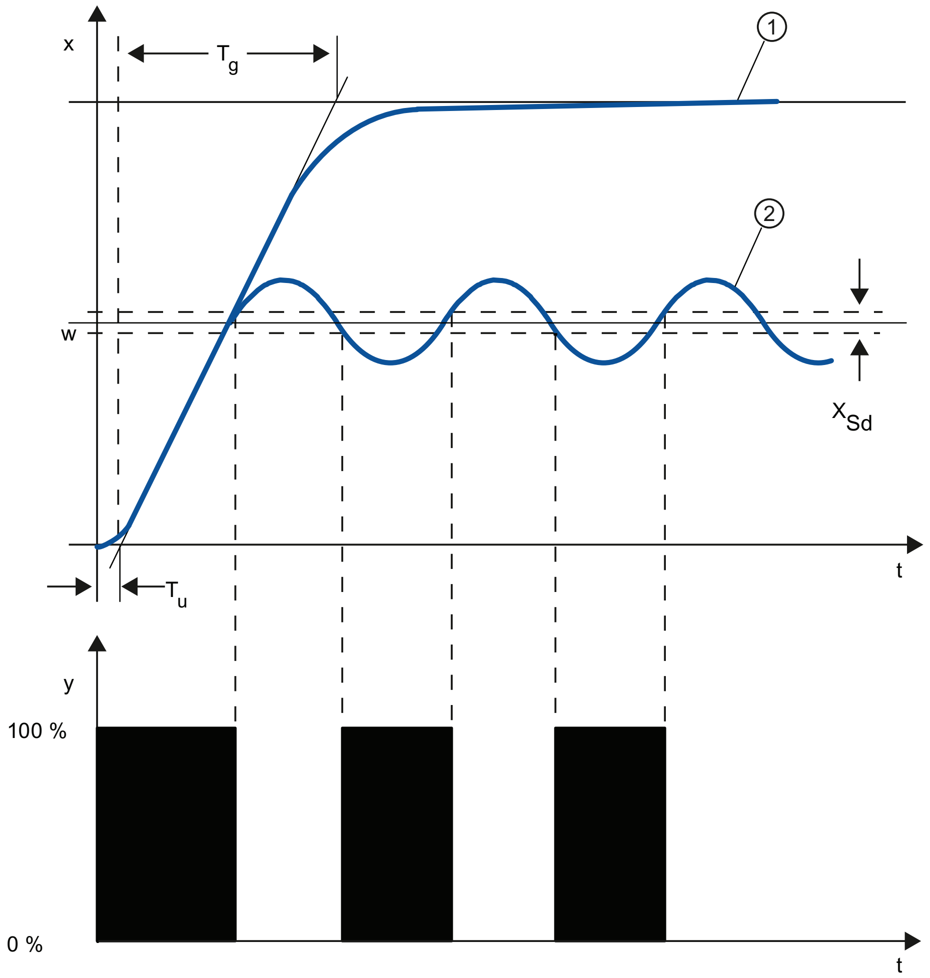
The amplitude and duration of the oscillation increase in proportion to the ratio between the delay time T_u and recovery time T_g of the controlled system. These controllers are used mainly for simple temperature control systems (such as electrically directly heated furnaces) or as limit-value signaling units.

The following diagram shows the characteristic of a two-step controller



①	ON
②	OFF
Y_h	Control range
w	Setpoint

The following diagram shows the control function of a two-step controller



①	Response characteristic without controller
②	Response characteristic with two-step controller
T_u	Delay time
T_g	Recovery time
X_{Sd}	Switching difference

Two-step controllers with feedback

The behavior of two-step controllers in the case of controlled systems with larger delay times, such as furnaces where the functional space is separated from the heating, can be improved by the use of electronic feedback.

The feedback is used to increase the switching frequency of the controller, which reduces the amplitude of the process value. In addition, the control-action results can be improved substantially in dynamic operation. The limit for the switching frequency is set by the output level. It should not exceed 1 to 5 switches per minute at mechanical actuators, such as relays and contactors. In the case of voltage and current outputs with downstream thyristor or Triac controllers high switching frequencies can be selected that exceed the limit frequency of the controlled system by far.

Since the switching pulses can no longer be determined at the output of the controlled system, results comparable with those of continuous controllers are obtained.

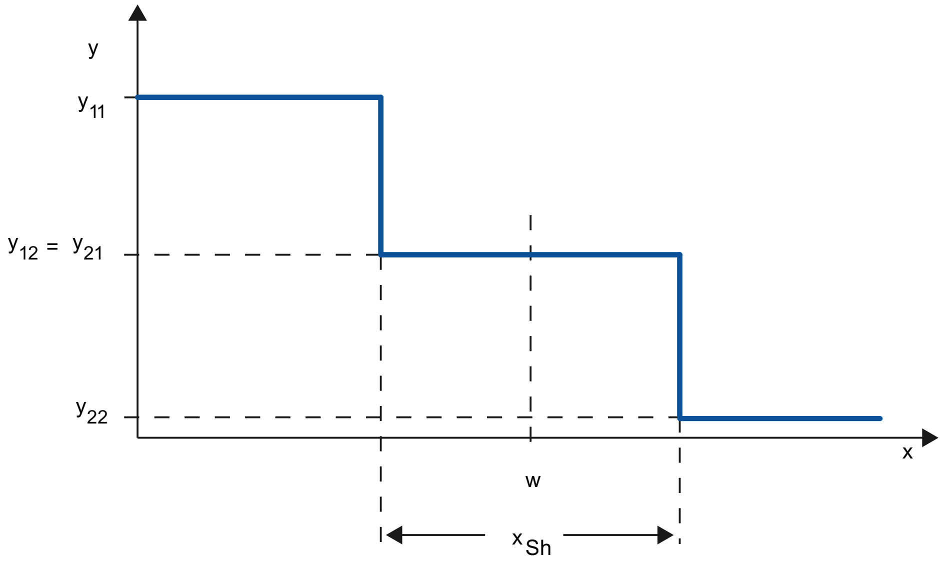
The output value is generated by pulse-width modulation of the output value of a continuous controller.

Two-step controllers with feedback are used for temperature control in furnaces, at processing machines in the plastics, textile, paper, rubber and foodstuff industries as well as for heating and cooling devices.

Three-step controllers

Three-step controllers are used for heating / cooling. These controllers have two switching points as their output. The control-action results are optimized through electronic feedback structures. Fields of applications for such controllers are heating, low-temperature, climatic chambers and tool heating units for plastic-processing machines.

The following diagram shows the characteristic of a three-step controller



y Output value, e.g.
y11 = 100% heating
y12 = 0% heating
y21 = 0% cooling
y22 = 100% cooling

x Physical quantity of the process value, e.g., temperature in° C

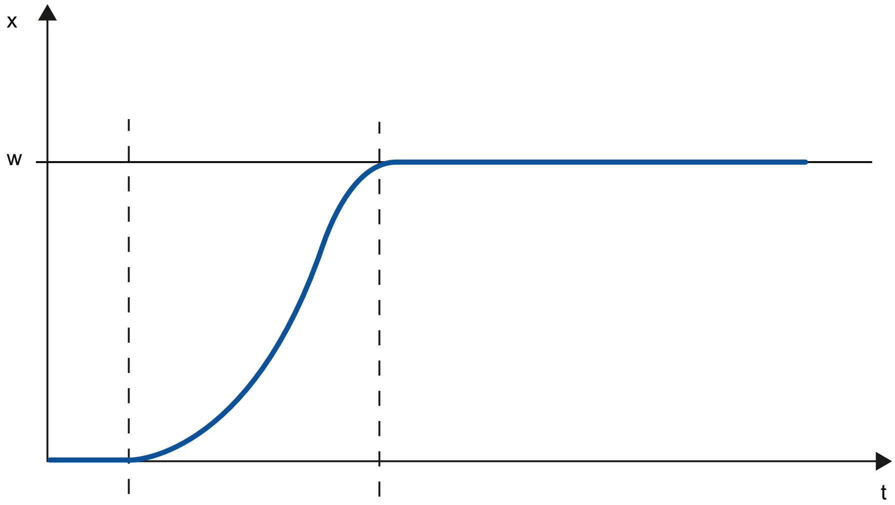
w Setpoint

x_{Sh} Distance between Switching Point 1 and Switching Point 2

11.1.1.5 Response to setpoint changes and disturbances

Response to setpoint changes

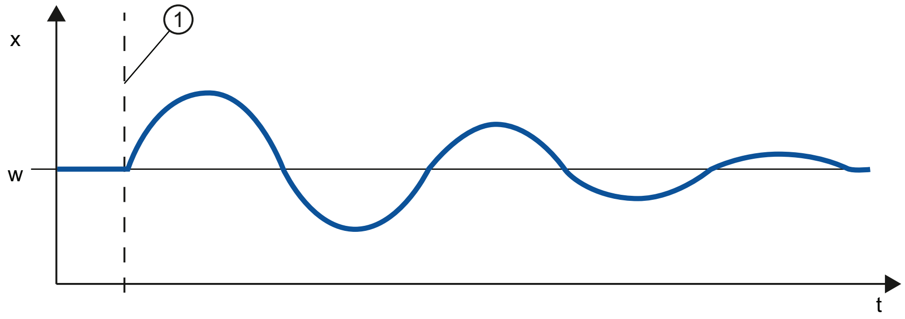
The process value should follow a setpoint change as quickly as possible. The response to setpoint changes is improved by minimizing fluctuation of the process value and the time required to reach the new setpoint.



x	Process value
w	Setpoint

Response to disturbances

The setpoint is influenced by disturbance variables. The controller has to eliminate the resulting control deviations in the shortest time possible. The response to disturbances is improved by minimizing fluctuation of the process value and the time required to reach the new setpoint.



x	Process value
w	Setpoint
①	Influencing a disturbance variable

Disturbance variables are corrected by a controller with integral action. A persistent disturbance variable does not reduce control quality because the control deviation is relatively constant. Dynamic disturbance variables have a more significant impact on control quality because of control deviation fluctuation. The control deviation is eliminated again only by means of the slow acting integral action.

A measurable disturbance variable can be included in the controlled system. This inclusion would significantly accelerated the response of the controller.

11.1.1.6 Control Response at Different Feedback Structures

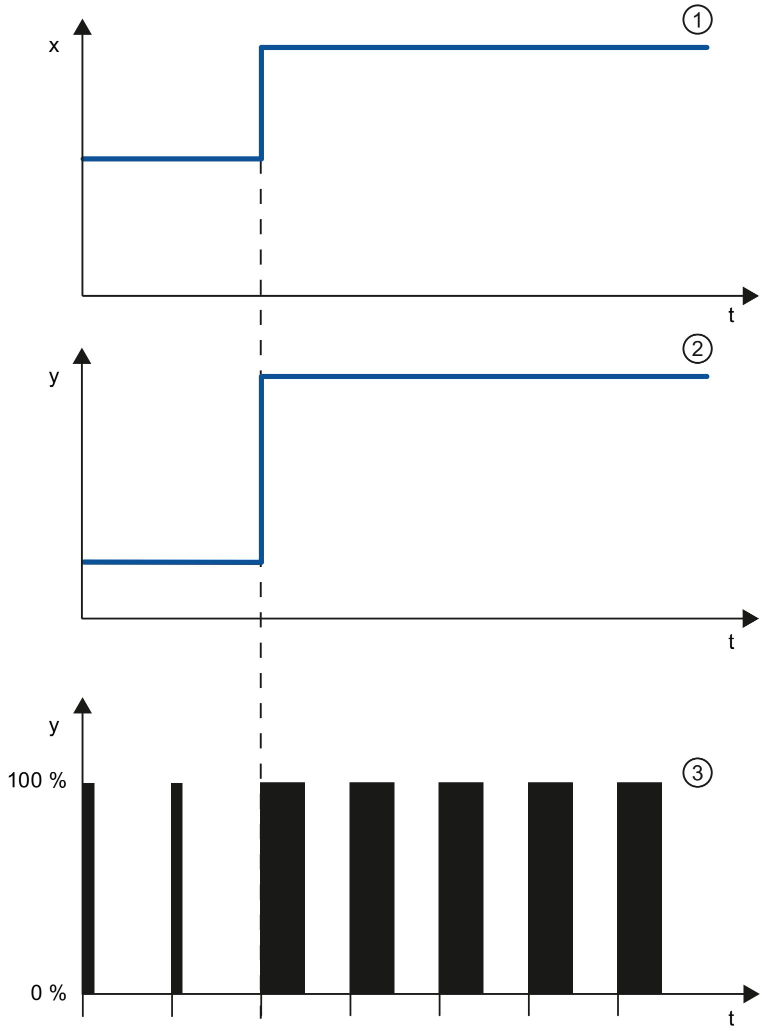
Control behavior of controllers

A precise adaptation of the controller to the time response of the controlled system is decisive for the controller's precise settling to the setpoint and optimum response to disturbance variables.

The feedback circuit can have a proportional action (P), proportional-derivative action (PD), proportional-integral action (PI), or proportional-integral-derivative action (PID).

If step functions are to be triggered by control deviations, the step responses of the controllers differ depending on their type.

Step response of a proportional action controller



① Control deviation
② Output value of a continuous controller
③ Output value of a pulse controller

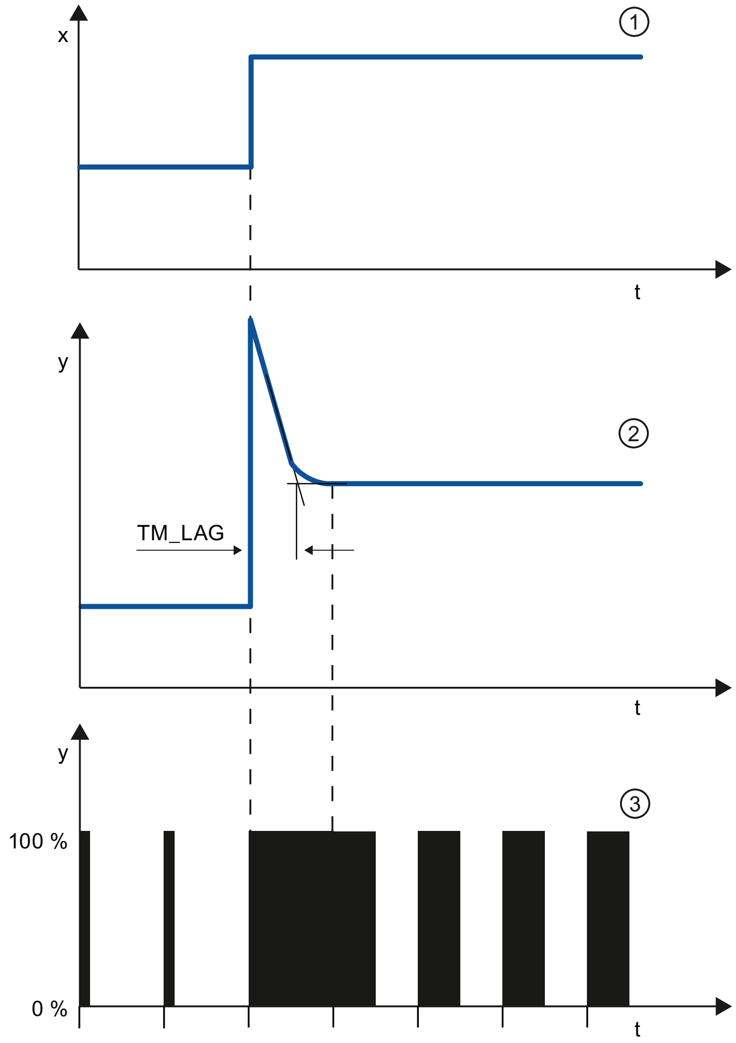
Equation for proportional action controller

Output value and control deviation are directly proportional, meaning:

Output value = proportional gain × control deviation

y = GAIN × x

Step response of a PD-action controller



①	Control deviation
②	Output value of a continuous controller
③	Output value of a pulse controller
TM_LAG	Delay of the Derivative action

Equation for PD-action controller

The following applies for the step response of the PD-action controller in the time range:

$$y = \text{GAIN} \cdot X_W \cdot \left(1 + \frac{\text{TD}}{\text{TM\_LAG}} \cdot e^{\frac{-t}{\text{TM\_LAG}}} \right)$$

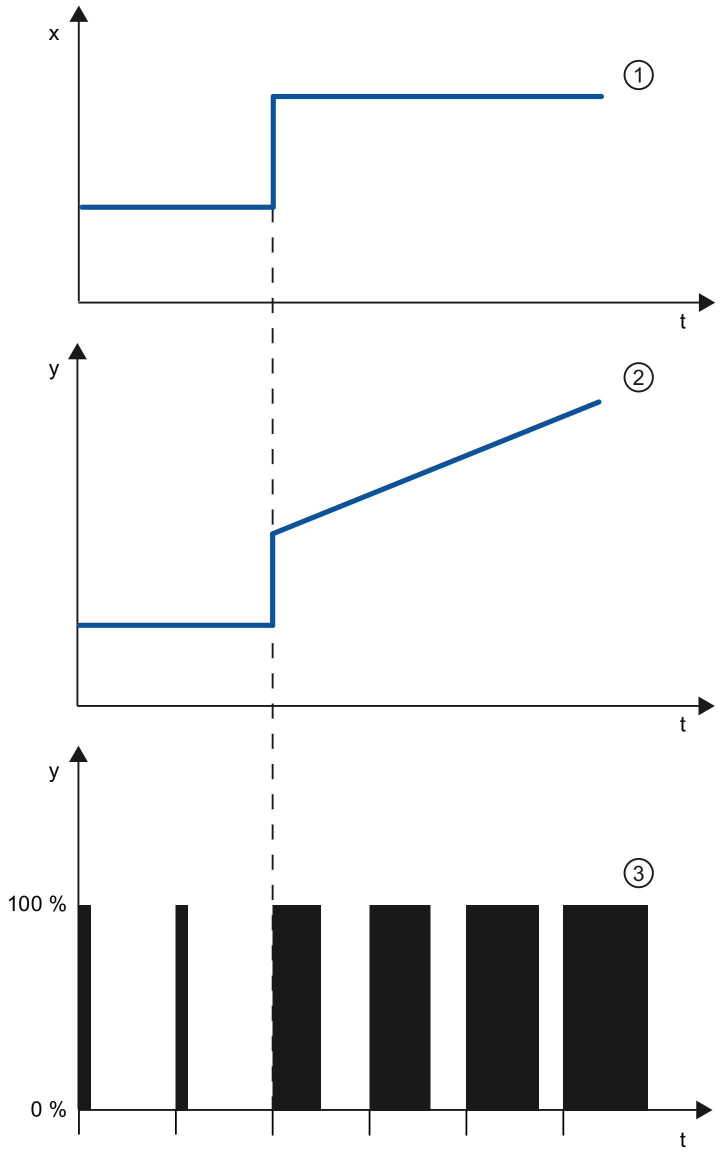
t = time interval since the step of the control deviation

The derivative action generates a output value as a function of the rate of change of the process value. A derivative action by itself is not suitable for controlling because the output value only follows a step of the process value. As long as the process value remains constant, the output value will no longer change.

The response to disturbances of the derivative action is improved in combination with a proportional action. Disturbances are not corrected completely. The good dynamic response is advantageous. A well attenuated, non-oscillating response is achieved during approach and setpoint change.

A controller with derivative action is not appropriate if a controlled system has pulsing measured quantities, for example, in the case of pressure or flow control systems.

Step response of a PI-action controller



① Control deviation
② Output value of a continuous controller
③ Output value of a pulse controller

An integral action in the controller adds the control deviation as a function of the time. This means that the controller corrects the system until the control deviation is eliminated. A sustained control deviation is generated at controllers with proportional action only. This effect can be eliminated by means of an integral action in the controller.

In practical experience, a combination of the proportional, integral and derivative actions is ideal, depending on the requirements placed on the control response. The time response of the individual components can be described by the controller parameters proportional gain GAIN, integral action time TI (integral action), and derivative action time TD (derivative action).

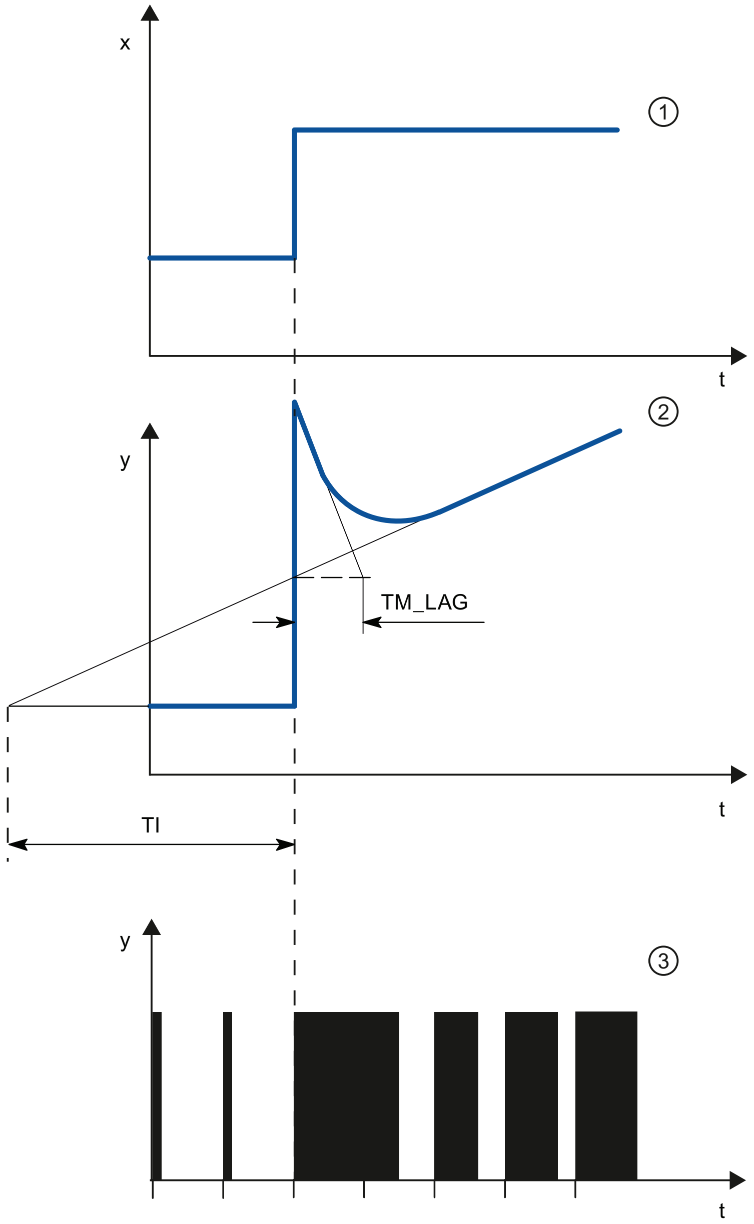
Equation for PI-action controller

The following applies for the step response of the PI-action controller in the time range:

$$y = GAIN \cdot X_W \cdot \left(1 + \frac{1}{TI \cdot t}\right)$$

t = time interval since the step of the control deviation

Step response of a PID controller



①	Control deviation
②	Output value of a continuous controller
③	Output value of a pulse controller
TM_LAG	Delay of the Derivative action
T_i	Integral action time

Equation for PID controller

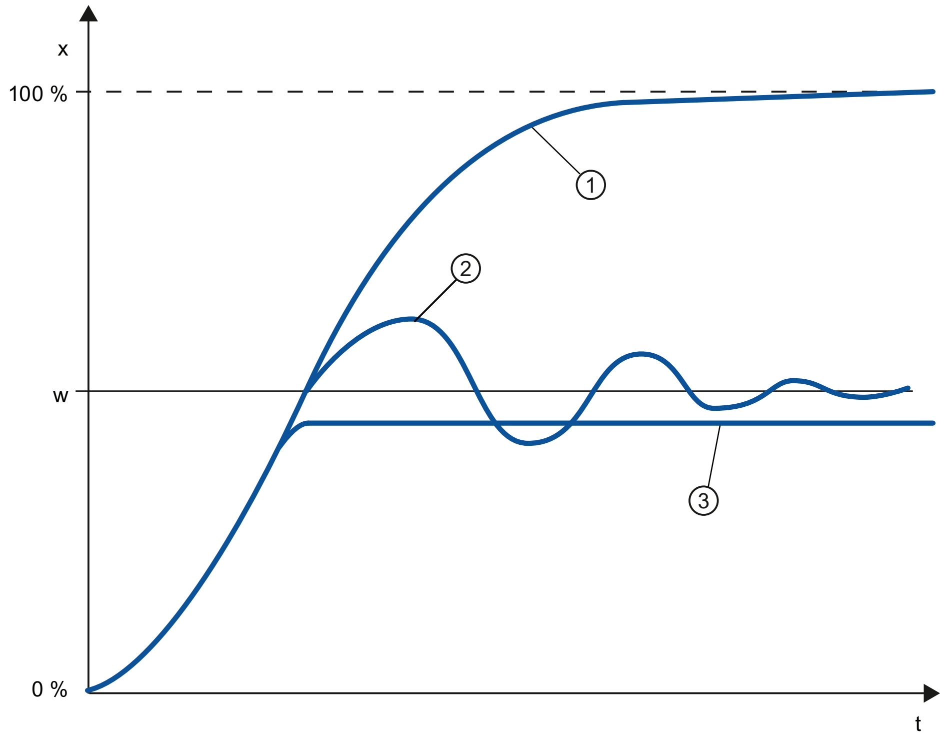
The following applies for the step response of the PID controller in the time range:

$$y = \mathrm{GAIN} \cdot X_w \cdot \left(1 + \frac{1}{\mathrm{TI} \cdot t} + \frac{\mathrm{TD}}{\mathrm{TM\_LAG}} \cdot e^{\frac{-t}{\mathrm{TM\_LAG}}}\right)$$

t = time interval since the step of the control deviation

Response of a controlled system with different controller structures

Most of the controller systems occurring in process engineering can be controlled by means of a controller with PI-action response. In the case of slow controlled system with a large dead time, for example temperature control systems, the control result can be improved by means of a controller with PID action.



①	No controller
②	PID controller
③	PD-action controller
w	Setpoint
x	Process value

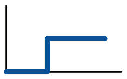
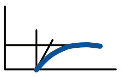
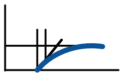
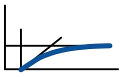
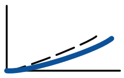
Controllers with PI and PID action have the advantage that the process value does not have any deviation from the setpoint value after settling. The process value oscillates over the setpoint during approach.

11.1.1.7 Selection of the controller structure for specified controlled systems

Selection of the Suitable Controller Structures

To achieve optimum control results, select a controller structure that is suitable for the controlled system and that you can adapt to the controlled system within specific limits.

The table below provides an overview of suitable combinations of a controller structure and controlled system.

Controlled system		Controller structure			
		P	PD	PI	PID
	With dead time only	Unsuitable	Unsuitable	Suitable	Unsuitable
	PT1 with dead time	Unsuitable	Unsuitable	Well suited	Well suited
	PT2 with dead time	Unsuitable	Suited conditionally	Well suited	Well suited
	Higher order	Unsuitable	Unsuitable	Suited conditionally	Well suited
	Not self-regulating	Well suited	Well suited	Well suited	Well suited

The table below provides an overview of suitable combinations of a controller structure and physical quantity.

Physical quantity	Controller structure			
	P	PD	PI	PID
	Sustained control deviation		No sustained control deviation	
Temperature	For low performance requirements and proportional action controlled systems with T_u/Tg < 0,1	Well suited	The most suitable controller structures for high performance requirements (except for specially adapted special controllers)	
Pressure	Suitable, if the delay time is inconsiderable	Unsuitable	The most suitable controller structures for high performance requirements (except for specially adapted special controllers)	
Flow rate	Unsuitable, because required GAIN range is usually too large	Unsuitable	Suitable, but integral action controller alone often better	Hardly required

11.1.1.8 PID parameter settings

Rule of Thumb for the Parameter Setting

Controller structure	Setting
P	GAIN ≈ v_{max} × T_u [° C]
PI	GAIN ≈ 1.2 × v_{max} × T_u [° C]
PD	GAIN ≈ 0.83 × v_{max} × T_u [° C] TD ≈ 0.25 × v_{max} × T_u [min] TM_LAG ≈ 0.5 × TD[min]
PID	GAIN ≈ 0.83 × v_{max} × T_u [° C] TI ≈ 2 × T_u [min] TD ≈ 0.4 × T_u [min] TM_LAG ≈ 0.5 × TD[min]
PD/PID	GAIN ≈ 0.4 × v_{max} × T_u [° C] TI ≈ 2 × T_u [min] TD ≈ 0.4 × T_u [min] TM_LAG ≈ 0.5 × TD[min]

Instead of $v_{max} = \Delta_x / \Delta_t$, you can use X_{max} / T_g.

In the case of controllers with PID structure the setting of the integral action time and differential-action time is usually coupled with each other.

The ratio TI / TD lies between 4 and 5 and is optimal for most controlled systems.

Non-observance of the differential-action time TD is uncritical at PD controllers.

In the case of PI and PID controllers, control oscillations occur if the integral action time TI has been select by more than half too small.

An integral action time that is too large slows down the settling times of disturbances. One cannot expect that the control loops operate "optimally" after the first parameter settings. Experience shows that adjusting is always necessary, when a system exists that is "difficult to control" with $T_u / T_g > 0.3$.

11.1.2 Configuring a software controller

11.1.2.1 Steps for the configuration of a software controller

Introduction

For the configuration of a software controller, you need an instruction with the control algorithm and a technology object. The technology object for a software controller corresponds with the instance DB of the instruction. The configuration of the controller is saved in the technology object. In contrast to the instance DBs of other instructions, technology objects are not stored by the program resources, but rather under PLC > Technology objects.

Overview of the technology objects

CPU	Instruction	Technology object	Description
S7-1200	PID_Compact	PID_Compact	Universal PID controller with integrated tuning
S7-1200	PID_3Step	PID_3Step	PID controller with integrated tuning for valves
S7-300/400	CONT_C	CONT_C	Continuous controller
S7-300/400	CONT_S	CONT_S	Step-by-step controller for actuators with integrating behavior
S7-300/400	TCONT_CP	TCONT_CP	Continuous temperature controller with pulse generator
S7-300/400	TCONT_S	TCONT_S	Temperature controller for actuators with integrating behavior
S7-300/400	PID_CP	PID_CP	Continuous controller with pulse generator (option package)
S7-300/400	PID_ES	PID_ES	Step-by-step controller for actuators with integrating behavior (option package)
S7-300/400	TUN_EC	TUN_EC	Optimization of a continuous controller (option package)
S7-300/400	TUN_ES	TUN_ES	Optimization of a step-by-step controller (option package)

All SW-controllers configure according to the same scheme:

Step	Description
1	Add technology object (Page 2882)
2	Configure technology object (Page 2883)
3	Call instruction in the user program (Page 2884)
4	Loading to CPU (Page 2885)
5	Commissioning software controller (Page 2885)
6	Save optimized PID_Parameter in the project (Page 2886)

See also

Display instance DB of a technology object. (Page 2887)

11.1.2.2 Add technology objects

Add technology object in the project navigator

When a technology object is added, an instance DB is created for the instruction of this technology object. The configuration of the technology object is stored in this instance DB.

Requirement

A project with a CPU has been created.

Procedure

To add a technology object, proceed as follows:

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Double-click "Add new object".
   The "Add new object" dialog box opens.
4. Click on the "PID" button.
   All available PID-controllers for this CPU are displayed.
5. Select the instruction for the technology object, for example, PID_Compact.
6. Enter an individual name for the technology object in the "Name" input field.
7. Select the "Manual" option if you want to change the suggested data block number of the instance DB.
8. Click "Further information" if you want to add own information to the technology object.
9. Confirm with "OK".

Result

The new technology object has been created and stored in the project tree in the "Technology objects" folder. The technology object is used if the instruction for this technology object is called in a cyclic interrupt OB.

Note

You can select the "Add new and open" check box at the bottom of the dialog box. This opens the configuration of the technology object after adding has been completed.

11.1.2.3 Configure technology objects

The properties of a technology object on a S7-1200 CPU can be configured in two ways.

- In the Inspector window of the programming editor
- In the configuration editor

The properties of a technology object on a S7-300/400 CPU can only be configured in the configuration editor.

Inspector window of the programming editor

In the Inspector window of the programming editor you can only configure the parameters required for operation.

The offline values of the parameters are also shown in online mode. You can only change the online values in the commissioning window.

To open the Inspector window of the technology object, follow these steps:

1. Open the "Program blocks" folder in the project tree.
2. Double click the block (cyclic interrupt OB) in which you open the instruction of the SW-controller.
   The block is opened in the work area.
3. Click on the instruction of the SW-controller.
4. In the Inspector window, select the "Properties" and "Configuration" tabs consecutively.

Configuration window

For each technology object, there is a specific configuration window in which you can configure all properties.

To open the configuration window of the technology object, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click the "Configuration" object.

Symbols

Icons in the area navigation of the configuration and in the Inspector window show additional details about the completeness of the configuration:

✓	**The configuration contains default values and is complete**. The configuration exclusively contains default values. With these default values the use of the technology object is possible without further changes.
✓	**The configuration contains values defined by the user and is complete** All input fields of the configuration contain valid values and at least one default setting was changed.
✗	**The configuration is incomplete or faulty** At least one input field or a collapsible list contains no or one invalid value. The corresponding field or the drop-down list box has a red background. When clicked the roll-out error message indicates the cause of the error.

The properties of a technology object are described in detail in the chapter for the technology object.

11.1.2.4 Call instruction in the user program

The instruction of the software controller must be called in a cyclic interrupt OB. The sampling time of the software controller is determined by the interval between the calls in the cyclic interrupt OB.

Requirement

The cyclic interrupt OB is created and the cycle time of the cyclic interrupt OB is correctly configured.

Procedure

Proceed as follows to call the instruction in the user program:

1. Open the CPU folder in the project tree.
2. Open the "Program blocks" folder.
3. Double-click the cyclic interrupt OB.
   The block is opened in the work area.
4. Open the "Technology" group in the "Instructions" window and the "PID Control" folder.
   The folder contains all instructions for software controllers that can be configured on the CPU.
5. Select the instruction and drag it to your cyclic interrupt OB.
   The "Call options" dialog box opens.
6. Select a technology object or type the name for a new technology object from the "Name" list.

Result

If the technology object does not exist yet, it is added. The instruction is added in the cyclic interrupt OB. The technology object is assigned to this call of the instruction.

11.1.2.5 Commissioning software controller

Procedure

To open the "Commissioning" work area of the technology object, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click the "Commissioning" object.

The commissioning functions are specific for each controller and are described there.

11.1.2.6 Loading to CPU

A new or modified configuration of the technology object must be downloaded to the CPU for the online mode. The following menu and shortcut menu commands are available for the download:

- Menu command Online > Download to device

  Downloads the configuration of the technology object, the compiled hardware data and the remaining software project data to the device.

- Menu command Online > Extended download to device

  Sets up an online connection to the selected device and downloads the compiled hardware and software project data, including the configuration of a technology object to the device.

The CPU object was selected in the project navigator

- Download to Device > All shortcut menu command

  Downloads the configuration of the technology object, the compiled hardware data and the remaining software project data to the device.

- Download to Device > Software shortcut menu command

  Downloads the modified configuration of the technology object and the modified blocks to a device. Only the modified objects are transferred to the device.

- Download to Device > Software (all blocks) shortcut menu command

  Downloads all blocks and the technology objects, including the objects which were not changed to the device.

The "Program blocks" object was selected in the project navigator.

- Download to Device > Software shortcut menu command

  Downloads the modified configuration of the technology object and the modified blocks to a device. Only the modified objects are transferred to the device.

- Download to Device > Software (all blocks) shortcut menu command

  Downloads all blocks and the technology objects, including the objects which were not changed to the device.

11.1.2.7 Save optimized PID parameter in the project

The software controller is optimized in the CPU. Through this, the values in the instance-DB on the CPU no longer agree with those in the project.

To update the PID parameter in the project with the optimized PID parameters, proceed as follows:

Requirement

- An online connection to the CPU is established and the CPU is in "RUN" mode.
- The functions of the commissioning window have been enabled by means of the "Start" button.

Procedure

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Open a technology object.
4. Double click on "Commissioning".
5. Click on the icon "Upload PID parameters".
6. Save the project.

Result

The currently active PID parameters are stored in the project data. When reloading the project data in the CPU, the optimized parameters are used.

11.1.2.8 Display instance DB of a technology object.

An instance DB, in which the parameter and static variables are saved, is created for each technology object.

Procedure

To display the instance DB of a technology object, proceed as follows:

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Highlight a technology object.
4. Select the command "Open DB editor" in the shortcut menu.

11.1.3 PID control (S7-1200)

11.1.3.1 Using PID_Compact

Technology object PID Compact

The "PID_Compact" technology object provides a continuous PID controller with integrated tuning. You can alternatively configure a pulse controller. Both manual and automatic mode are possible.

The PID controller of a controlled system continuously acquires the measured process value and compares it with the desired setpoint. From the resulting control deviation, the instruction PID_Compact calculates an output value through which the process value is compared with as quickly and stable as possible with the setpoint. The output value for the PID controller consists of three actions:

- P action

  The proportional action of the output value increases in proportion to the control deviation.

- I action

  The integral action of the output value increases until the control deviation has been balanced.

- D action

  The derivative action increases with the rate of change of control deviation. The process value is corrected to the setpoint as quickly as possible. The derivative action will be reduced again if the rate of change of control deviation drops.

The instruction PID_Compact calculates the proportional, integral and derivative parameters for your controlled system during "pretuning". "Fine tuning" can be used to tune the parameters further. You do not need to manually determine the parameters.

Additional information

- Steps for the configuration of a software controller (Page 2881)
- Add technology objects (Page 2882)
- Configure technology objects (Page 2883)
- Configuring PID Compact (Page 2888)

Configuring PID Compact

Basic settings

Introduction

Configure the following properties of the "PID_Compact" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset
- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)

Setpoint, process value and output value

You can only configure the setpoint, process value and output value in the Inspector window of the programming editor. Select the source for each value:

- Instance DB

  The value saved in the instance DB is used.

  Value must be updated in the instance DB by the user program.

  There should be no value at the instruction.

  Change via HMI possible.

- Instruction

  The value connected to the instruction is used.
  The value is written to the instance DB each time the instruction is called.

  No change via HMI possible.

Controller type

Physical quantity

Select the unit of measurement and physical quantity for the setpoint and process value in the "Controller type" group. The setpoint and process value will be displayed in this unit.

Control logic

Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

Start-up behavior after reset

To change straight to the last active mode after restarting the CPU, select the "Enable last mode after CPU restart" check box.

PID_Compact will remain in "Inactive" mode if the check box is cleared.

Setpoint

Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.

   Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

Process value

PID_Compact will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

Output value

PID_Compact offers three output values. Your actuator will determine which output value you use.

- Output_PER

  The actuator is triggered via an analog output and controlled with a continuous signal, e.g. 0...10V, 4...20mA.
- Output

  The output value needs to be processed by the user program, for example because of nonlinear actuator response.
- Output_PWM

  The actuator is controlled via a digital output. Pulse width modulation creates minimum ON and minimum OFF times.

Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output_PER (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to process the output value using the user program:

1. Select the entry "Output" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the name of the variable you are using to process the output value.
4. Transfer the processed output value to the actuator via a digital or analog CPU output.

Proceed as follows to use the digital output value:

1. Select the entry "Output_PWM" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the digital output.

Process value settings

Configure the scaling of your process value and specify the process value absolute limits In the "Process value settings" configuration window.

Scaling the process value

If you have configured the use of Input_PER in the basic settings, you will need to convert the value of the analog input into the physical quantity of the process value. The current configuration will be displayed in the Input_PER display.

Input_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

1. Enter the low pair of values in the "Scaled low process value" and "Low" input fields.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are saved in the hardware configuration. Proceed as follows to use the value pairs from the hardware configuration:

1. Select the instruction PID_Compact in the programming editor.
2. Connect Input_PER with an analog input in the basic settings.
3. Click on the "Automatic setting" button in the process value settings.

The existing values will be overwritten with the values from the hardware configuration.

Monitoring process value

Specify the absolute high and low limit of the process value. As soon as these limits are violated during operation, the controller switches off and the output value is set to 0%. You must enter reasonable limits for your controlled system. Reasonable limits are important during optimization to obtain optimal PID parameters.

The default for the "High limit process value" is 120 %. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer reported for a violation of the "High limit process value". Only a wire-break and a short-circuit are recognized and the PID_Compact switches to "Inactive" mode.

⚠ WARNING

If you set very high process value limits (for example -3.4*10^{38}...+3.4*10^{38}), process value monitoring will be disabled. Your system may then be damaged if an error occurs.

See also

Monitoring process value (Page 2892)

PWM limits (Page 2893)

Output value limits (Page 2894)

PID parameters (Page 2894)

Advanced settings

Monitoring process value

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID_Compact instruction:

- At the InputWarning_H output parameter if the warning high limit has been exceeded
- At the InputWarning_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

Example

Process value high limit = 98° C; warning high limit = 90° C

Warning low limit = 10° C; process value low limit = 0° C

PID_Compact will respond as follows:

Process value	InputWarning_H	InputWarning_L	Operating mode
> 98° C	TRUE	FALSE	Inactive
≤ 98° C and > 90° C	TRUE	FALSE	Automatic mode
≤ 90° C and ≥ 10° C	FALSE	FALSE	Automatic mode
< 10° C and ≥ 0° C	FALSE	TRUE	Automatic mode
< 0° C	FALSE	TRUE	Inactive

See also

Process value settings (Page 2891)

PWM limits (Page 2893)

Output value limits (Page 2894)

PID parameters (Page 2894)

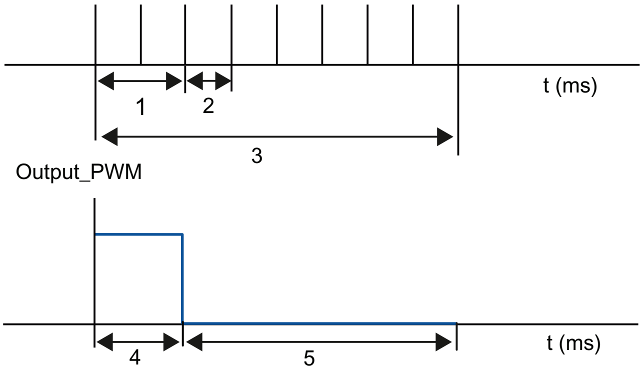
PWM limits

In the "PWM limits" configuration window, configure the valid minimum ON and OFF times for the actuator. Extend the minimum ON and OFF times if you want to reduce the switching frequency.

The configuration of the PWM limitations affects output parameter "Output_PWM". The output value is switched to "Output_PWM" by means of pulse-width modulation. The output value is scaled internally to a value from 0.0 to 100.0%.

If you are using the output value outputs "Output" or "Output_PER", you must configure the value 0.0 for the minimum ON and OFF times.

The figure below illustrates the correlation between the sampling time PID_Compact, the sampling time PID_Algorithm, the minimum on/off times, as well as the resultant pulse duration/interval.



1	Minimum ON time	r_Lmn_Pwm_PPTm
2	Sampling time PID_Compact	r_Cycle
	Minimum OFF time	r_Lmn_Pwm_PBTm
3	Sampling time PID algorithm	r_Ctrl_Cycle
4	Minimum pulse duration	r_Lmn_Pwm_PPTm
5	Interval	n × r_Lmn_Pwm_PBTm

The "Minimum ON time" or "Minimum OFF time" corresponds to a multiple of the sampling time PID_Compact. The sampling time of the PID algorithm is set to the higher of the ON and OFF time values.

The smallest impulse time corresponds to the sampling time PID_Compact.

Note

The switching output for the pulse-width modulation is controlled by the "PID_Compact" instruction. The pulse generators integrated in the CPU are not used.

See also

Process value settings (Page 2891)

Monitoring process value (Page 2892)

Output value limits (Page 2894)

PID parameters (Page 2894)

Output value limits

In the "Output value limits" configuration window, configure the absolute limits of your output value in percent. Absolute output value limits are not violated in neither manual mode nor in automatic mode. If a output value outside the limits is specified in manual mode, the effective value is limited in the CPU to the configured limits.

The valid output value limit values depend on the Output used.

Output	-100.0 to 100.0
Output_PER	-100.0 to 100.0
Output_PWM	0.0 to 100.0

PID_Compact sets the output value to 0.0 if an error occurs. 0.0 must therefore always be within the output value limits. You will need to add an offset to Output and Output_PER in the user program if you want an output value low limit of greater than 0.0.

See also

Process value settings (Page 2891)

Monitoring process value (Page 2892)

PWM limits (Page 2893)

PID parameters (Page 2894)

PID parameters

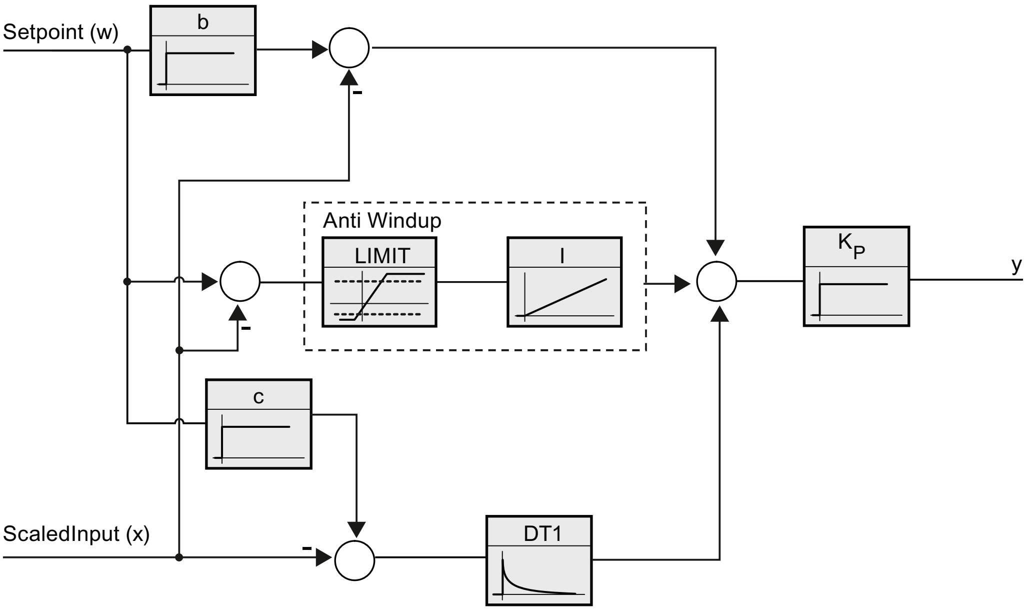
The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

The following equation is used to calculate the output value.

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
y	Output value
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient (T1 = a × T_D)
T_D	Derivative action time
c	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm:



Proportional gain

The value specifies the proportional gain of the controller.

Integral action time

The integral action time determines the time behavior of the integral action. The integral action is deactivated with integral action time = 0.0.

Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

Sampling time PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is rounded to a multiple of the sampling time PID_Compact. All other functions of PID_Compact are executed at every call.

Rule for tuning

Select whether PI or PID parameters are to be calculated in the "Controller structure" drop-down list.

- PID

  Calculates PID parameters during pretuning and fine tuning.
- PI

  Calculates PI parameters during pretuning and fine tuning.
- User-defined

  The drop-down list displays "User-defined" if you have configured different controller structures for pretuning and fine tuning via a user program.

Commissioning PID Compact

Commissioning

The commissioning window helps you commission the PID controller. You can monitor the values for the setpoint, process value and output value along the time axis in the trend view. The following functions are supported in the commissioning window:

- Controller pretuning
- Controller fine tuning

  Use fine tuning for fine adjustments to the PID parameters.
- Monitoring the current closed-loop control in the trend view
- Testing the controlled system by specifying a manual output value

All functions require an online connection to the CPU to have been established.

Basic handling

- Select the desired sampling time in the "Sampling time" drop-down list.

  All values in the commissioning window are updated in the selected update time.
- Click the "Start" icon in the measuring group if you want to use the commissioning functions.

  Value recording is started. The current values for the setpoint, process value and output value are entered in the trend view. Operation of the commissioning window is enabled.
- Click the "Stop" icon if you want to end the commissioning functions.

  The values recorded in the trend view can continue to be analyzed.

Closing the commissioning window will terminate recording in the trend view and delete the recorded values.

See also

Pretuning (Page 2898)

Fine tuning (Page 2899)

Using the trend view (Page 2901)

"Manual" mode (Page 2904)

Save optimized PID parameter in the project (Page 2905)

Pretuning

The pretuning function determines the process response to a setpoint jump and scans the system for the point of inflection. The optimized PID parameters are calculated as a function of the maximum slope and dead time of the controlled system.

The higher the stability of the process value, the easier it is to calculate the PID parameters and increase precision of the result. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. The PID parameters are backed up before being recalculated.

Requirement

- The "PID_Compact" instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- PID_Compact is in "inactive" or "manual" mode.
- The setpoint may not be changed during controller tuning. PID_Compact will otherwise be deactivated.
- The setpoint and the process value lie within the configured limits (see "Process value monitoring" configuration).
- The difference between setpoint and process value is greater than 30% of the difference between process value high limit and process value low limit.
- The distance between the setpoint and the process value > 50% of the setpoint.

Procedure

Proceed as follows to carry out "pretuning":

1. Double-click on "PID_Compact > Commissioning" in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list.
3. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - Pretuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

Note

Click the "Stop" icon when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

The PID parameters will have been optimized if pretuning has been executed without errors. PID_Compact changes to automatic mode and uses the optimized parameters. The optimized PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID_Compact will change to "Inactive" mode.

See also

Parameters State and sRet.i_Mode (Page 1781)

Commissioning (Page 2897)

Fine tuning (Page 2899)

Using the trend view (Page 2901)

"Manual" mode (Page 2904)

Save optimized PID parameter in the project (Page 2905)

Fine tuning

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are optimized for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated on the basis of the findings. PID parameters from fine tuning usually have better master control and disturbance behavior than PID parameters from pretuning.

PID_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

Requirement

- The PID_Compact instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- The setpoint and the process value lie within the configured limits (see "Process value monitoring" configuration).
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- The setpoint may not be changed during controller tuning.
- PID_Compact is in inactive mode, automatic mode or manual mode.

Process depends on initial situation

Fine tuning can be started in "inactive", "automatic" or "manual" mode. Fine tuning proceeds as follows when started in:

- Automatic mode

  Start fine tuning in automatic mode if you wish to improve the existing PID parameters using controller tuning.

  PID_Comact will regulate using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.

- Inactive or manual mode

  If the requirements for pretuning are met, pretuning is started. The PID parameters established will be used for adjustment until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. If pretuning is not possible, PID_Compact will change to "Inactive" mode.

  An attempt is made to reach the setpoint with a minimum or maximum output value if the process value for pretuning is already too near the setpoint. This can produce increased overshoot.

Procedure

Proceed as follows to carry out "fine tuning":

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - The process of fine tuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

Note

Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

The PID parameters will have been optimized if fine tuning has been executed without errors. PID_Compact changes to automatic mode and uses the optimized parameters. The optimized PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during "fine tuning", PID_Compact will change to "inactive" mode.

See also

Parameters State and sRet.i_Mode (Page 1781)

Commissioning (Page 2897)

Pretuning (Page 2898)

Using the trend view (Page 2901)

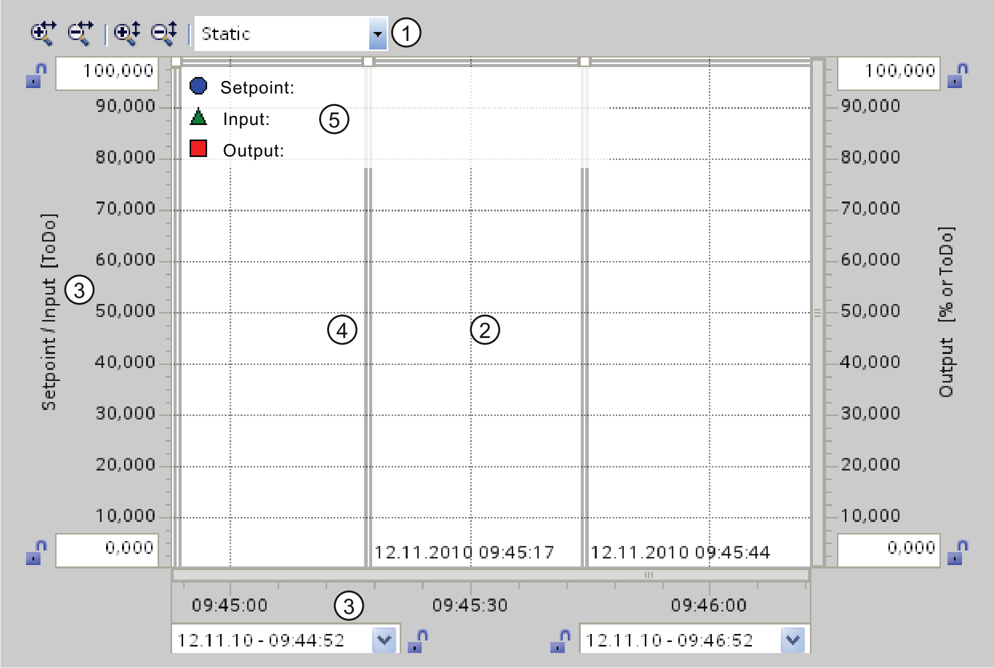
"Manual" mode (Page 2904)

Save optimized PID parameter in the project (Page 2905)

Using the trend view

The trend view is used for graphic illustration of the setpoint, process value and output value over time. The values of the trend view are updated in the selected sampling time. The recording of the trend values starts when the "Start" button is clicked and ends when the "Stop" button is clicked.

Elements of the trend view



①	Selection of the display mode
②	Trend view
③	Area for moving and scaling the axes
④	Ruler
⑤	Legend with the trend values at the ruler

Selection of the display mode

The following display modes can be selected for displaying the trend recording:

- Strip (continuous display)

  New trend values are entered at the right-hand trend view. Previous trend values are scrolled to the left. The time axis cannot be moved.

- Scope (jumping area display)

  New trend values are entered within the trend view from left to right. When the right-hand margin of the trend view is reached, the monitoring area is moved one view width to the right. The time axis can be moved within the limits of the monitoring area.

- Sweep (rotating display)

  New trend values are displayed in the trend view in accordance with a rotating display. The trend values are entered from left to right. The trend values of the last rotating display are overwritten at the writing position. The time axis cannot be moved.

- Static (static area display)

  Writing of the trends is interrupted. Recording of the new trend values is carried out in the background. The time axis can be moved across the entire previous recording period.

If the recording is stopped, you can view the total recorded trend direction.

In the "Static" display mode, you can move the visible range of the ratio display along the axis by dragging it with the mouse.

Trend view

The trends for the setpoint (Setpoint), process value (Input) and output value (Output) are displayed in the trend view. In addition to different colors, the trends are identified by symbols (refer to legend).

Moving and scaling the axes

The axes for the setpoint and process value, for the output value and the time axis can be moved and scaled individually. The right and left mouse buttons are assigned alternating functions. You can use the following icons and mouse actions:

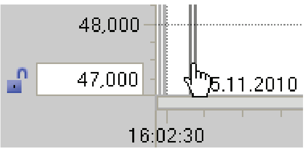
↕	Moving the setpoint / process value axes or the output value axis up or down. The axis can only be shifted when no scaling point of the axis is locked.
↔	Moving the time axis to the right or left. The axis can only be shifted when no scaling point of the axis is locked.
⊕↕ ⊖↕	Stretch and compress the setpoint, process value axes or the output value axis. • The scaling of the axes is stretched or compressed symmetrically if none of the scaling values is blocked. • Blocked scaling values are retained when you stretch or compress an axis.
⊕↔ ⊖↔	Stretching and compressing the time axis • The scaling of the axis is stretched or compressed symmetrically if none of the scaling values is blocked. • Blocked scaling values are retained when you stretch or compress an axis.
▲	Stretching and compressing the setpoint / process value axes or the output value axis The low scaling value is not changed as a result of stretching or compressing.
▼	Stretching and compressing the setpoint / process value axes or the output value axis. The high scaling value is not changed during stretching or compressing.
◀	Stretching and compressing the time axis. The right scaling value is not changed as a result of stretching or compressing.
▶	Stretching and compressing the time axis. The left scaling value is not changed as a result of stretching or compressing.
🔓 50,000	Enter a scaling value.
🔒 50,000	The current scaling value can be blocked using the padlock icon. Both values can be locked by an axis.

	Double-clicking in the trend view optimizes the scaling and position of the setpoint, process value and output value in the trend view.
	Double-clicking in area of the setpoint / process value axis or the output value axis restored the default position and scaling of the axis.

Working with rulers

Use one or several rulers to analyze discrete values of the trend profile.

The ruler parking position is located at the left edge of the trend area. A second parking position is located at the top edge of the trend area. However, the values of these rulers cannot be displayed.



Move the mouse to the left edge of the trend area and watch out for the transition of the mouse pointer. Drag a vertical ruler onto the measuring trend that you want to analyze. To insert further rulers, press CTRL, click on the ruler and drag the new ruler into the trend area.

The trend values are output left-aligned on the ruler. The time of the ruler position is displayed at the base of the ruler.

The trend values of the active ruler are displayed in the legend. If several rulers are dragged to the trend area, the respectively last ruler is active. The active ruler is indicated by the correspondingly colored ┳ symbol.

You can reactivate an inactive ruler by clicking it.

Use the shortcut ALT+Click to remove rulers which are no longer required.

See also

Commissioning (Page 2897)

Pretuning (Page 2898)

Fine tuning (Page 2899)

"Manual" mode (Page 2904)

Save optimized PID parameter in the project (Page 2905)

"Manual" mode

The following section describes how you can use the "Manual" operating mode in the commissioning window of the "PID Compact" technology object.

Requirement

- The "PID_Compact" instruction is called in a cyclic interrupt OB.
- An online connection to the CPU has been established and the CPU is in the "RUN" mode.
- The functions of the commissioning window have been enabled via the "Start" icon.

Procedure

Use "Manual mode" in the commissioning window if you want to test the process by specifying a manual value. To define a manual value, proceed as follows:

1. Select the check box "Manual mode" in the "Online status of the controller" area.

   PID_Compact operates in manual mode. The most recent current output value remains in effect.
2. Enter the manual value in the "Output" field as a % value.
3. Click the control icon .

Result

The manual value is written to the CPU and immediately goes into effect.

Note

PID_Compact continues to monitor the process value. If the process value limits are exceeded, PID_Compact is deactivated.

Clear the "Manual mode" check box if the output value is to be specified again by the PID controller. The change to automatic mode is bumpless.

See also

Parameters State and sRet.i_Mode (Page 1781)

Commissioning (Page 2897)

Pretuning (Page 2898)

Fine tuning (Page 2899)

Using the trend view (Page 2901)

Save optimized PID parameter in the project (Page 2905)

Save optimized PID parameter in the project

The software controller is optimized in the CPU. Through this, the values in the instance-DB on the CPU no longer agree with those in the project.

To update the PID parameter in the project with the optimized PID parameters, proceed as follows:

Requirement

- An online connection to the CPU is established and the CPU is in "RUN" mode.
- The functions of the commissioning window have been enabled by means of the "Start" button.

Procedure

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Open a technology object.
4. Double click on "Commissioning".
5. Click on the icon "Upload PID parameters".
6. Save the project.

Result

The currently active PID parameters are stored in the project data. When reloading the project data in the CPU, the optimized parameters are used.

See also

Commissioning (Page 2897)

Pretuning (Page 2898)

Fine tuning (Page 2899)

Using the trend view (Page 2901)

"Manual" mode (Page 2904)

11.1.3.2 Using PID_3Step

Technology object PID_3Step

The technology object PID_3Step provides a PID controller with tuning for valves or actuators with integral response.

You can configure the following controllers:

- Three-point step controller with position feedback
- Three-point step controller without position feedback
- Valve controller with analog output value

PID_3Step continuously acquires the measured process value within a control loop and compares it with the setpoint. From the resulting control deviation, PID_3Step calculates an output value through which the process value reaches the setpoint as quickly and steadily as possible. The output value for the PID controller consists of three actions:

- P action

  The proportional action of the output value increases in proportion to the control deviation.

- I action

  The integral action of the output value increases until the control deviation has been balanced.

- D action

  The derivative action increases with the rate of change of control deviation. The process value is corrected to the setpoint as quickly as possible. The derivative action will be reduced again if the rate of change of control deviation drops.

The instruction PID_3Step calculates the proportional, integral and derivative parameters for your controlled system during pretuning. Fine tuning can be used to tune the parameters further. You do not need to manually determine the parameters.

Additional information

- Steps for the configuration of a software controller (Page 2881)
- Add technology objects (Page 2882)
- Configure technology objects (Page 2883)
- Configuring PID_3Step (Page 2908)

Configuring PID_3Step

Basic settings

Introduction

Configure the following properties of the "PID_3Step" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset
- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)
- Position feedback (only in the Inspector window)

Setpoint, process value, output value and position feedback

You can only configure the setpoint, process value, output value and position feedback in the Inspector window of the programming editor. Select the source for each value:

- Instance DB

  The value saved in the instance DB is used.

  Value must be updated in the instance DB by the user program.

  There should be no value at the instruction.

  Change via HMI possible.

- Instruction

  The value connected to the instruction is used.
  The value is written to the instance DB each time the instruction is called.

  No change via HMI possible.

Controller type

Physical quantity

Select the unit of measurement and physical quantity for the setpoint and process value in the "Controller type" group. The setpoint and process value will be displayed in this unit.

Control logic

Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

Start-up behavior after reset

To change straight to the last active mode after restarting the CPU, select the "Enable last mode after CPU restart" check box.

PID_3Step will remain in "Inactive" mode if the check box is cleared.

Setpoint

Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.

   Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

Process value

PID_3Step will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

Position feedback

Position feedback configuration depends upon the actuator used.

- Actuator without position feedback
- Actuator with digital endstop signals
- Actuator with analog position feedback
- Actuator with analog position feedback and endstop signals

Actuator without position feedback

Proceed as follows to configure PID_3Step for an actuator without position feedback:

1. Select the entry "No Feedback" in the drop-down list "Feedback".

Actuator with digital endstop signals

Proceed as follows to configure PID_3Step for an actuator with endstop signals:

1. Select the entry "No Feedback" in the drop-down list "Feedback".
2. Activate the "Actuator endstop signals" check box.
3. Select "Instruction" as source for Actuator_H and Actuator_L.
4. Enter the addresses of the digital inputs for Actuator_H and Actuator_L.

Actuator with analog position feedback

Proceed as follows to configure PID_3Step for an actuator with analog position feedback:

1. Select the entry "Feedback" or "Feedback_PER" in the drop-down list "Feedback".
   - Use the analog input value for Feedback_PER. Configure Feedback_PER scaling in the actuator settings.
   - Process the analog input value for Feedback using your user program.
2. Select "Instruction" as source.
3. Enter the address of the analog input or the variable of your user program.

Actuator with analog position feedback and endstop signals

Proceed as follows to configure PID_3Step for an actuator with analog position feedback and endstop signals:

1. Select the entry "Feedback" or "Feedback_PER" in the drop-down list "Feedback".
2. Select "Instruction" as source.
3. Enter the address of the analog input or the variable of your user program.
4. Activate the "Actuator endstop signals" check box.
5. Select "Instruction" as source for Actuator_H and Actuator_L.
6. Enter the addresses of the digital inputs for Actuator_H and Actuator_L.

Output value

PID_3Step offers an analog output value (Output_PER) and digital output values (Output_UP, Output_DN). Your actuator will determine which output value you use.

- Output_PER

  The actuator is triggered via an analog output and controlled with a continuous signal, e.g. 0...10V, 4...20mA.
- Output_UP, Output_DN

  The actuator is controlled via two digital outputs.

Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to use the digital output value:

1. Select the entry "Output (digital)" in the drop-down list "Output".
2. Select "Instruction" for Output_UP and Output_DN.
3. Enter the addresses of the digital outputs.

Proceed as follows to process the output value using the user program:

1. Select the entry corresponding to the actuator in the drop-down list "Output".
2. Select "Instruction".
3. Enter the name of the variable you are using to process the output value.
4. Transfer the processed output value to the actuator via a digital CPU output.

Process value settings

Configure the scaling of your process value and specify the process value absolute limits In the "Process value settings" configuration window.

Scaling the process value

If you have configured the use of Input_PER in the basic settings, you will need to convert the value of the analog input into the physical quantity of the process value. The current configuration will be displayed in the Input_PER display.

Input_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

1. Enter the low pair of values in the "Scaled low process value" and "Low" input fields.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are saved in the hardware configuration. Proceed as follows to use the value pairs from the hardware configuration:

1. Select the instruction PID_3Step in the programming editor.
2. Connect Input_PER to an analog input in the basic settings.
3. Click on the "Automatic setting" button in the process value settings.

   The existing values will be overwritten with the values from the hardware configuration.

Monitoring process value

Specify the absolute high and low limit of the process value. As soon as these limits are violated during operation, the controller switches off and the output value is set to 0%. You must enter reasonable limits for your controlled system. Reasonable limits are important during optimization to obtain optimal PID parameters. The default for the "High limit process value" is 120 %. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer reported for a violation of the "High limit process value". Only a wire-break and a short-circuit are recognized and the PID_3Step behaves as configured in the responses in the event of an error.

⚠ WARNING

If you set very high process value limits (for example -3.4*10^{38}...+3.4*10^{38}), process value monitoring will be disabled. Your system may then be damaged if an error occurs.

Actuator settings

Actuator-specific times

Configure the motor transition time and the minimum ON and OFF times to prevent damage to the actuator. You can find the specifications in the actuator data sheet.

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. The actuator is moved for a maximum of 110% of the motor transition time in one direction. You can measure the motor transition time during commissioning.

If you are using "Output_UP" or "Output_DN", you can reduce the switching frequency with the minimum on and minimum OFF time.

The on or off times calculated are totaled in automatic mode and only become effective when the sum is greater than or equal to the minimum on or OFF time.

A rising edge at Manual_UP or Manual_DN in manual mode will operate the actuator for at least the minimum on or OFF time.

Reaction to errors

PID_3Step is preset so that the controller stays active in most cases in the event of an error. If errors occur frequently in controller mode, this default reaction has a negative effect on the control response. In this case, check the Errorbits parameter and eliminate the cause of the error.

PID_3Step generates a programmable output value in response to an error:

- Current value

  PID_3Step is switched off and no longer modifies the actuator position.
- Current value for error while error is pending

  The controller functions of PID_3Step are switched off and the position of the actuator is no longer changed.

If the following errors occur in automatic mode, PID_3Step returns to automatic mode as soon as the errors are no longer pending.

- 0002h: Invalid value at parameter Input_PER.
- 0200h: Invalid value at parameter Input.
- 0800h: Sampling time error
- 1000h: Invalid value at parameter Setpoint.
- 2000h: Invalid value at parameter Feedback_PER.
- 4000h: Invalid value at parameter Feedback.
- 8000h: Error in digital position feedback.

If one of these error occurs in manual mode, PID_3Step remains in manual mode.

If an error occurs during the tuning or transition time measurement, PID_3Step is switched off.

- Substitute output value

  PID_3Step moves the actuator to the substitute output value and then switches off.

- Substitute output value while error is pending

  PID_3Step moves the actuator to the substitute output value. When the substitute output value is reached, PID_3Step reacts as it does with "Current value for while error is pending".

Enter the substitute output value in "%".

Only substitute output values 0% and 100% can be approached precisely in the case of actuators without analog position feedback. The actuator is moved in one direction at 110% of the motor transition time to ensure the high or low endstop is reached. There endstop signals take priority. A substitute output value not equal to 0% or 100% is approached via an internally simulated position feedback. This procedure does not, however, allow the exact approach of substitute output value.

All substitute output values can be approached precisely with actuators with analog position feedback.

Scaling position feedback

If you have configured the use of Feedback_PER in the basic settings, you will need to convert the value of the analog input into %. The current configuration will be displayed in the "Feedback" display.

Feedback_PER is scaled using a low and high value pair.

1. Enter the low pair of values in the "Low endstop" and "Low" input boxes.
2. Enter the high pair of values in the "High endstop" and "High" input boxes.

"Low endstop" must be less than "High endstop"; "Low" must be less than "High".

The valid values for "High endstop" and "Low endstop" depend upon:

- No Feedback, Feedback, Feedback_PER
- Output (analog), Output (digital)

Output	Feedback	Low endstop	High endstop
Output (digital)	No Feedback	0.0% cannot be set	100.0% cannot be set
Output (digital)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (digital)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	No Feedback	0.0% cannot be set	100.0% cannot be set
Output (analog)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%

Limiting the output value

Absolute output value limits are not violated in neither manual mode nor in automatic mode. If an output value outside the limits is specified in manual mode, the effective value will be limited in the CPU to the configured limits. PID_3Step must be able to close the valve completely. Therefore, zero must be included in the output value limits.

Enter the absolute output value limits in the "Output value high limit" and "Output value low limit" input boxes. The output value limits must be within "Low endstop" and "High endstop".

If Feedback is available and Output (digital) is set, you cannot limit the output value. The digital outputs are reset with Actuator_H = TRUE or Actuator_L = TRUE, or after a travel time amounting to 110% of the motor transition time.

Advanced settings

Monitoring process value

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID_3Step instruction:

- At the InputWarning_H output parameter if the warning high limit has been exceeded
- At the InputWarning_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

Example

Process value high limit = 98° C; warning high limit = 90° C

Warning low limit = 10° C; process value low limit = 0° C

PID_3Step will respond as follows:

Process value	InputWarning_H	InputWarning_L	Operating mode
> 98° C	TRUE	FALSE	Inactive
≤ 98° C and > 90° C	TRUE	FALSE	Automatic mode
≤ 90° C and ≥ 10° C	FALSE	FALSE	Automatic mode
< 10° C and ≥ 0° C	FALSE	TRUE	Automatic mode
< 0° C	FALSE	TRUE	Inactive

PID parameters

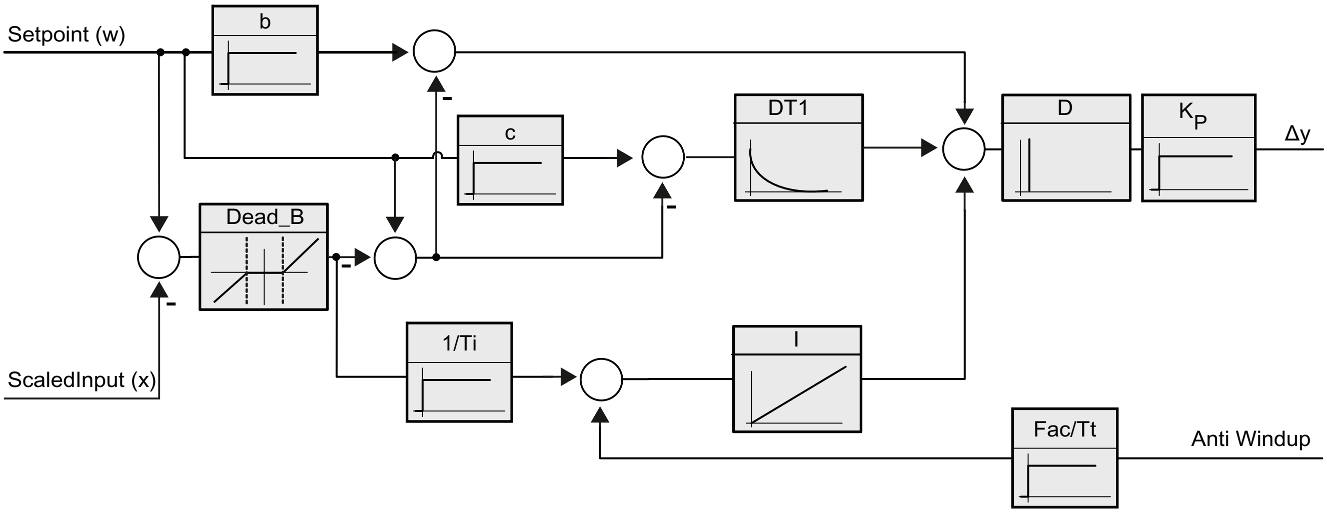
The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

The following equation is used to calculate the output value.

$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
y	Output value
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient ($T1 = a \times T_D$)
T_D	Derivative action time
c	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm::



Proportional gain

The value specifies the proportional gain of the controller.

Integral action time

The integral action time determines the time behavior of the integral action. The integral action is deactivated with integral action time = 0.0.

Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

Sampling time PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is rounded to a multiple of the sampling time PID_3Step. All other functions of PID_3Step are executed at every call.

Deadband width

The deadband suppresses the noise component in the steady controller state. The deadband width specifies the size of the deadband. The deadband is off if the deadband width is 0.0.

Commissioning PID_3Step

Commissioning

You can monitor the setpoint, process value and output value over time in the "Tuning" working area. The following commissioning functions are supported in the curve plotter:

- Controller pretuning
- Controller fine tuning
- Monitoring the current closed-loop control in the trend view

All functions require an online connection to the CPU to have been established.

Basic handling

- Select the desired sampling time in the "Sampling time" drop-down list.

  All values in the tuning working area are updated in the selected update time.
- Click the "Start" icon in the measuring group if you want to use the commissioning functions.

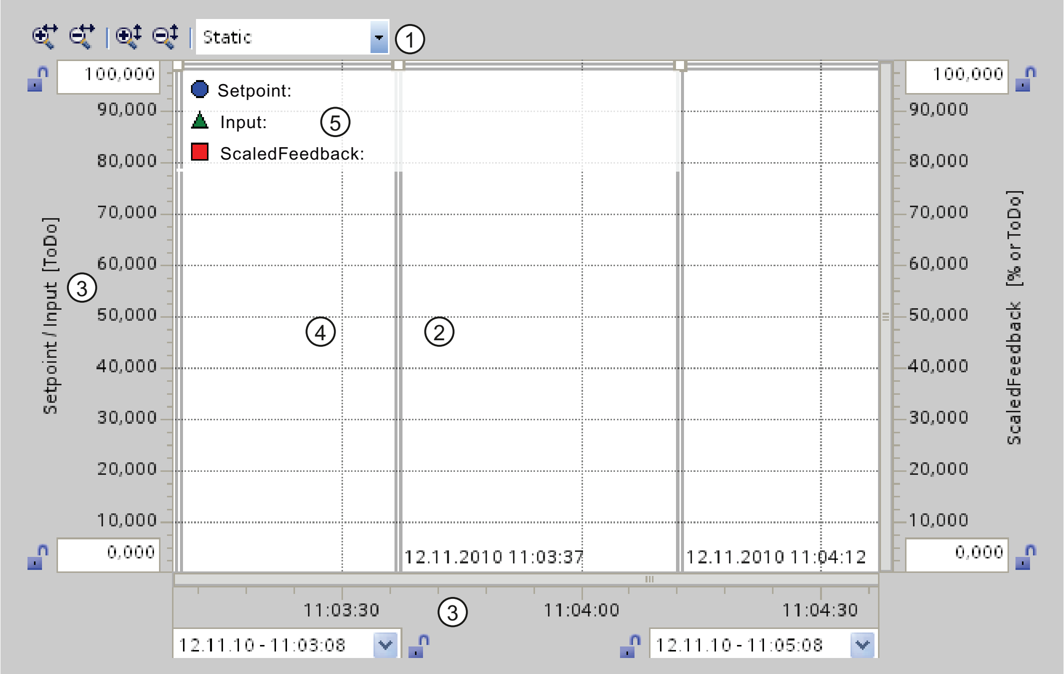
  Value recording is started. The current values for the setpoint, process value and output value are entered in the trend view. Operation of the commissioning window is enabled.
- Click the "Stop" icon if you want to end the commissioning functions.

  The values recorded in the trend view can continue to be analyzed.
- Closing the commissioning window will terminate recording in the trend view and delete the recorded values.

Using the trend view

The trend view is used for graphic illustration of the setpoint, process value and output value over time. The values of the trend view are updated in the selected sampling time. The recording of the trend values starts when the "Start" button is clicked and ends when the "Stop" button is clicked.

Elements of the trend view



①	Selection of the display mode
②	Trend view
③	Area for moving and scaling the axes
④	Ruler
⑤	Legend with the trend values at the ruler

Selection of the display mode

The following display modes can be selected for displaying the trend recording:

- Strip (continuous display)

  New trend values are entered at the right-hand trend view. Previous trend values are scrolled to the left. The time axis cannot be moved.

- Scope (jumping area display)

  New trend values are entered within the trend view from left to right. When the right-hand margin of the trend view is reached, the monitoring area is moved one view width to the right. The time axis can be moved within the limits of the monitoring area.

- Sweep (rotating display)

  New trend values are displayed in the trend view in accordance with a rotating display. The trend values are entered from left to right. The trend values of the last rotating display are overwritten at the writing position. The time axis cannot be moved.

- Static (static area display)

  Writing of the trends is interrupted. Recording of the new trend values is carried out in the background. The time axis can be moved across the entire previous recording period.

If the recording is stopped, you can view the total recorded trend direction.

In the "Static" display mode, you can move the visible range of the ratio display along the axis by dragging it with the mouse.

Trend view

The trends for the setpoint (Setpoint), process value (Input) and output value are displayed in the trend view. In addition to different colors, the trends are identified by symbols (refer to legend).

Moving and scaling the axes

The axes for the setpoint and process value, for the output value and the time axis can be moved and scaled individually. The right and left mouse buttons are assigned alternating functions. You can use the following icons and mouse actions:

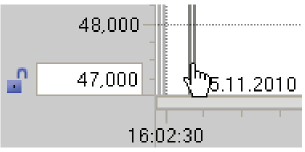
⇕	Moving the setpoint / process value axes or the output value axis up or down. The axis can only be shifted when no scaling point of the axis is locked.
⇔	Moving the time axis to the right or left. The axis can only be shifted when no scaling point of the axis is locked.
⊕↕ ⊖↕	Stretch and compress the setpoint, process value axes or the output value axis. • The scaling of the axes is stretched or compressed symmetrically if none of the scaling values is blocked. • Blocked scaling values are retained when you stretch or compress an axis.
⊕↔ ⊖↔	Stretching and compressing the time axis • The scaling of the axis is stretched or compressed symmetrically if none of the scaling values is blocked. • Blocked scaling values are retained when you stretch or compress an axis.
▲	Stretching and compressing the setpoint / process value axes or the output value axis The low scaling value is not changed as a result of stretching or compressing.
▼	Stretching and compressing the setpoint / process value axes or the output value axis. The high scaling value is not changed during stretching or compressing.
◀	Stretching and compressing the time axis. The right scaling value is not changed as a result of stretching or compressing.
▶	Stretching and compressing the time axis. The left scaling value is not changed as a result of stretching or compressing.
🔓 50,000	Enter a scaling value.
🔒 50,000	The current scaling value can be blocked using the padlock icon. Both values can be locked by an axis.

	Double-clicking in the trend view optimizes the scaling and position of the setpoint, process value and output value in the trend view.
	Double-clicking in area of the setpoint / process value axis or the output value axis restored the default position and scaling of the axis.

Working with rulers

Use one or several rulers to analyze discrete values of the trend profile.

The ruler parking position is located at the left edge of the trend area. A second parking position is located at the top edge of the trend area. However, the values of these rulers cannot be displayed.



Move the mouse to the left edge of the trend area and watch out for the transition of the mouse pointer. Drag a vertical ruler onto the measuring trend that you want to analyze. To insert further rulers, press CTRL, click on the ruler and drag the new ruler into the trend area.

The trend values are output left-aligned on the ruler. The time of the ruler position is displayed at the base of the ruler.

The trend values of the active ruler are displayed in the legend. If several rulers are dragged to the trend area, the respectively last ruler is active. The active ruler is indicated by the correspondingly colored ╥ symbol.

You can reactivate an inactive ruler by clicking it.

Use the shortcut ALT+Click to remove rulers which are no longer required.

Pretuning

The pretuning function determines the process response to a setpoint jump and scans the system for the point of inflection. The optimized PID parameters are calculated as a function of the maximum slope and dead time of the controlled system.

The higher the stability of the process value, the easier it is to calculate the PID parameters and increase precision of the result. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. The PID parameters are backed up before being recalculated.

The setpoint is frozen during pretuning.

Requirement

- The PID_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- PID_3Step is in "inactive" or "manual" mode.
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).

Procedure

Proceed as follows to carry out pretuning:

1. Double-click on "PID_3Step > Commissioning" in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list in the working area "Tuning".
3. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - Pretuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

The PID parameters will have been optimized if pretuning has been executed without errors. PID_3Step changes to automatic mode and uses the optimized parameters. The optimized PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID_3Step changes to "Inactive" mode.

Fine tuning

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are optimized for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated on the basis of the findings. PID parameters from fine tuning usually have better master control and disturbance behavior than PID parameters from pretuning.

PID_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

The setpoint is frozen during fine tuning.

Requirement

- The PID_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- The motor transition time has been configured or measured.
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- PID_3Step is in inactive mode, automatic mode or manual mode.

Process depends on initial situation

Fine tuning proceeds as follows when started in:

- Automatic mode

  Start fine tuning in automatic mode if you wish to improve the existing PID parameters using controller tuning.

  PID_3Step will regulate using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.
- Inactive or manual mode

  Pretuning is always started first. The PID parameters established will be used for adjustment until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.

Procedure

Proceed as follows to carry out "fine tuning":

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - The process of fine tuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

The PID parameters will have been optimized if fine tuning has been executed without errors. PID_3Step changes to automatic mode and uses the optimized parameters. The optimized PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during fine tuning, PID_3Step will change to "inactive" mode.

Commissioning with manual PID parameters

Procedure

Proceed as follows to commission PID_3Step with manual PID parameters:

1. Double-click on "PID_3Step > Configuration" in the project tree.
2. Click on "Advanced settings > PID Parameters" in the configuration window.
3. Select the check box "Enable direct input".
4. Enter the PID parameters.
5. Double-click on "PID_3Step > Commissioning" in the project tree.
6. Establish an online connection to the CPU.
7. Load the PID parameters to the CPU.
8. Click on the "Activate controller" icon.

Result

PID_3Step changes to automatic mode and controls using the current PID parameters.

Measuring the motor transition time

Introduction

PID_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation are average values for this type of actuator. The value for the specific actuator used may differ.

You can measure the motor transition time during commissioning if you are using actuators with position feedback or endstop signals.

The motor transition time cannot be measured if neither position feedback nor endstop signals are available.

Actuators with analog position feedback

Proceed as follows to measure motor transition time with position feedback:

Requirement

- Feedback or Feedback_PER has been selected in the basic settings and the signal has been connected.
- An online connection to the CPU has been established.

1. Select the "Use position feedback" check box.
2. Enter where the actuator is to be moved to in the "Target position" input field.

   The current position feedback (starting position) will be displayed. The target position must be within the output value limits. The difference between "Target position" and "Current position feedback" must be at least 50% of the valid output value range.
3. Click the ▶ "Start transition time measurement" icon.

Result

The actuator is moved from the starting position to the target position. Time measurement starts immediately and ends when the actuator reaches the target position. The motor transition time is calculated according to the following equation:

Motor transition time = (output value high limit – output value low limit) × Measuring time / AMOUNT (target position – starting position).

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. PID_3Step will change to "Inactive" mode once transition time measurement is complete.

Note

Click on the icon "Load measured transition time" to load the motor transition time measured to the project.

Actuators with endstop signals

Proceed as follows to measure the transition time of actuators with endstop signals:

Requirement

- The "Endstop signals" check box in the basic settings has been selected and Actuator_H and Actuator_L are connected.
- An online connection to the CPU has been established.

Proceed as follows to measure motor transition time with endstop signals:

1. Select the "Use actuator endstop signals" check box.
2. Select the direction in which the actuator is to be moved.
   - Open - close - open

     The actuator is moved first to the high endstop, then to the low endstop and then back to the high endstop.
   - Close - open - close

     The actuator is moved first to the low endstop, then to the high endstop and then back to the low endstop.
3. Click the ▶ "Start transition time measurement" icon.

Result

The actuator is moved in the selected direction. Time measurement will start once the actuator has reached the first endstop and will end when the actuator reaches this endstop for the second time. The motor transition time is equal to the time measured divided by two.

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. PID_3Step will change to "Inactive" mode once transition time measurement is complete.

Cancelling transition time measurement

PID_3Step will change to "Inactive" mode immediately if you cancel transition time measurement. The actuator will stop being moved. You can reactive PID-3Step in the curve plotter.

11.2 Motion Control

11.2.1 Motion Control (S7-1200)

11.2.1.1 Using S7-1200 Motion Control

Introduction

Motion functionality of the CPU S7-1200

The TIA Portal, together with the "Motion Control" functionality of the CPU S7-1200, supports you in controlling stepper motors and servo motors with pulse interface:

- In the TIA Portal, you configure the "Axis" and "Command table" technology objects. The CPU S7-1200 controls the pulse and direction outputs for control of the drives using these technology objects.
- In the user program you control the axis by means of motion control instructions and initiate motion jobs of your drive.

See also

Hardware components for motion control (Page 2929)

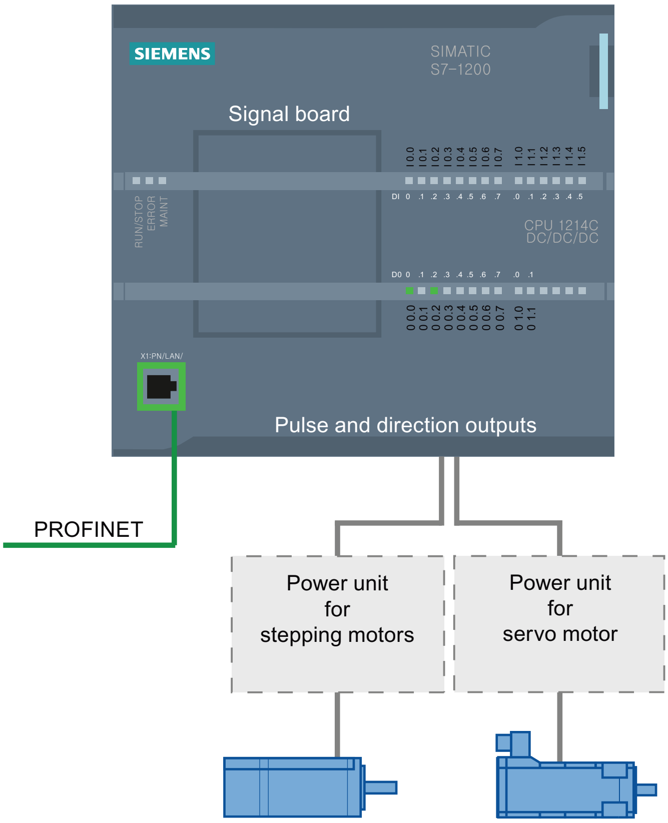
Integration of the axis technology object (Page 2941)

Use of the command table technology object (Page 2969)

Command table technology object tools (Page 2969)

Hardware components for motion control

The representation below shows the basic hardware configuration for a motion control application with the CPU S7-1200.



CPU S7-1200:

CPU S7-1200 combines the functionality of a programmable logic controller with motion control functionality for operation of stepper motors and servo motors with pulse interface. The motion control functionality takes over the control and monitoring of the drives.

The DC/DC/DC variants of the CPU S7-1200 have onboard outputs for direct control of drives. The relay variants of the CPU require one of the signal boards described below to control a drive.

Signal board

You add further inputs and outputs to the CPU with the signal boards. The digital outputs can be used as pulse and direction outputs for controlling drives as required.

In CPUs with relay outputs, the pulse signal cannot be output on the on-board outputs because the relays do not support the necessary switching frequencies. A signal board with digital outputs must be used to enable you to work with the PTO (Pulse Train Output) on these CPUs.

When a DC/DC/DC variant of the CPU S7-1200 is used together with a signal board, the maximum number of controllable drives is limited to "2".

PROFINET

Use the PROFINET interface to establish the online connection between the CPU S7-1200 and the programming device. In addition to the online functions of the CPU, additional commissioning and diagnostic functions are available for motion control.

Maximum number of controllable drives

The maximum number of controllable drives for the various CPU versions is provided in the following table:

CPU		Signal board					
		without	DI2/DO2 x DC24V 20kHz	DI2/DO2 x DC24V 200kHz	DO4 x DC24V 200kHz	DI2/DO2 x DC5V 200kHz	DO4 x DC5V 200kHz
CPU 1211C, CPU 1212C, CPU 1214C	DC/DC/DC	2	2	2	2	2	2
	AC/DC/RLY	-	1	1	2	1	2
	DC/DC/RLY	-	1	1	2	1	2

Limit frequencies of pulse outputs

The following limit frequencies apply to the pulse outputs:

Pulse output	Limit frequencies for technology object "Axis" V1.0	Limit frequencies for technology object "Axis" V2.0 and higher
Onboard	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 100 kHz
Signal board DI2/DO2 x DC24V 20kHz	2 Hz ≤ f ≤ 20 kHz	2 Hz ≤ f ≤ 20 kHz
Signal board DI2/DO2 x DC24V 200kHz	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 200 kHz
Signal board DO4 x DC24V 200kHz	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 200 kHz
Signal board DI2/DO2 x DC5V 200kHz	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 200 kHz
Signal board DO4 x DC5V 200kHz	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 200 kHz

Ordering information

The order information listed below applies to the currently installed product phase (without any installed Hardware Support Packages) of the TIA Portal.

Name	MLFB (order no.)
CPU 1211C DC/DC/DC	6ES7211-1AD30-0XB0
CPU 1211C AC/DC/RLY	6ES7211-1BD30-0XB0
CPU 1211C DC/DC/RLY	6ES7211-1HD30-0XB0
CPU 1212C DC/DC/DC	6ES7212-1AD30-0XB0
CPU 1212C AC/DC/RLY	6ES7212-1BD30-0XB0
CPU 1212C DC/DC/RLY	6ES7212-1HD30-0XB0
CPU 1214C DC/DC/DC	6ES7214-1AE30-0XB0
CPU 1214C AC/DC/RLY	6ES7214-1BE30-0XB0
CPU 1214C DC/DC/RLY	6ES7214-1HE30-0XB0
Signal board DI2/DO2 x DC24V 20kHz	6ES7223-0BD30-0XB0
Signal board DI2/DO2 x DC24V 200kHz	6ES7223-3BD30-0XB0
Signal board DO4 x DC24V 200kHz	6ES7222-1BD30-0XB0
Signal board DI2/DO2 x DC5V 200kHz	6ES7223-3AD30-0XB0
Signal board DO4 x DC5V 200kHz	6ES7222-1AD30-0XB0

Use a Hardware Support Package (HSP) to install new hardware components. The hardware component will then be available in the hardware catalog.

See also

Motion functionality of the CPU S7-1200 (Page 2928)

CPU outputs relevant for motion control (Page 2931)

Basics for working with S7-1200 Motion Control

CPU outputs relevant for motion control

Pulse and direction output

The CPU provides one pulse output and one direction output for controlling a stepper motor drive or a servo motor drive with pulse interface. The pulse output provides the drive with the pulses required for motor motion. The direction output controls the travel direction of the drive.

Pulse and direction outputs are permanently assigned to one another. Onboard CPU outputs or outputs of a signal board can be used as pulse and direction outputs. You select between onboard CPU outputs and outputs of the signal board during device configuration under Pulse generators (PTO/PWM) on the "Properties" tab.

The following table shows the address assignment of the pulse and direction outputs:

CPU S7-1200:	Without signal board				Signal boards DI2/DO2 *)				Signal boards DO4 **)			
	Outputs PTO1		Outputs PTO2		Outputs PTO1		Outputs PTO2		Outputs PTO1		Outputs PTO2	
	Pls.	Dir.	Pls.	Dir.	Pls.	Dir.	Pls.	Dir.	Pls.	Dir.	Pls.	Dir.
CPU 1211C, CPU 1212C, CPU 1214C (DC/DC/DC)	Ax.0	Ax.1	Ax.2	Ax.3	Ax.0 Ay.0	Ax.1 Ay.1	Ax.2	Ax.3	Ax.0 Ay.0	Ax.1 Ay.1	Ax.2 Ay.2	Ax.3 Ay.3
CPU 1211C, CPU 1212C, CPU 1214C (AC/DC/RLY)	-	-	-	-	Ay.0	Ay.1	-	-	Ay.0	Ay.1	Ay.2	Ay.3
CPU 1211C, CPU 1212C, CPU 1214C (DC/DC/RLY)	-	-	-	-	Ay.0	Ay.1	-	-	Ay.0	Ay.1	Ay.2	Ay.3

x = Initial byte address of onboard CPU outputs (default value = 0)

y = Initial byte address of signal board outputs (default value = 4)

* If a DC/DC/DC CPU variant is used together with a DI2/DO2 signal board, the signals of the PTO1 can be generated via the onboard CPU outputs or via the signal board.

** If a DC/DC/DC CPU variant is used together with a DO4 signal board, the signals for PTO1 and PTO2 can be generated via the onboard CPU outputs or via the signal board.

Drive interface

For motion control, you can optionally parameterize a drive interface for "Drive enabled" and "Drive ready". When using the drive interface the digital output for the drive enable and the digital input for "drive ready" can be freely selected.

Note

The firmware will take control via the corresponding pulse and direction outputs if the PTO (Pulse Train Output) has been selected and assigned to an axis.

With this takeover of the control function, the connection between the process image and I/O output is also disconnected. While the user has the possibility of writing the process image of pulse and direction outputs via the user program or watch table, this is not transferred to the I/O output. Accordingly, it is also not possible to monitor the I/O output via the user program or watch table. The information read reflects the value of the process image and does not match the real status of the I/O output.

For all other CPU outputs that are not used permanently by the CPU firmware, the status of the I/O output can be controlled or monitored via the process image, as usual.

See also

How the pulse interface works (Page 2933)

Relationship between the travel direction and voltage level at the direction output (Page 2934)

Hardware and software limit switches (Page 2935)

Jerk limit (Page 2936)

Homing (Page 2937)

Hardware components for motion control (Page 2929)

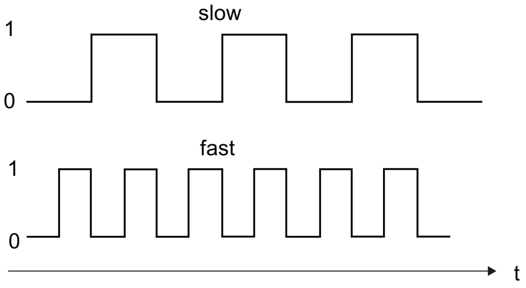
Integration of the axis technology object (Page 2941)

Tools of the axis technology object (Page 2944)

How the pulse interface works

Depending on the settings of the stepper motor, each pulse affects the movement of the stepper motor by a specific angle. If the stepper motor is set to 1000 pulses per revolution, for example, it moves 0.36° per pulse.

The speed of the stepper motor is determined by the number of pulses per time unit.



(The statements made here also apply to servo motors with pulse interface.)

See also

CPU outputs relevant for motion control (Page 2931)

Relationship between the travel direction and voltage level at the direction output (Page 2934)

Hardware and software limit switches (Page 2935)

Jerk limit (Page 2936)

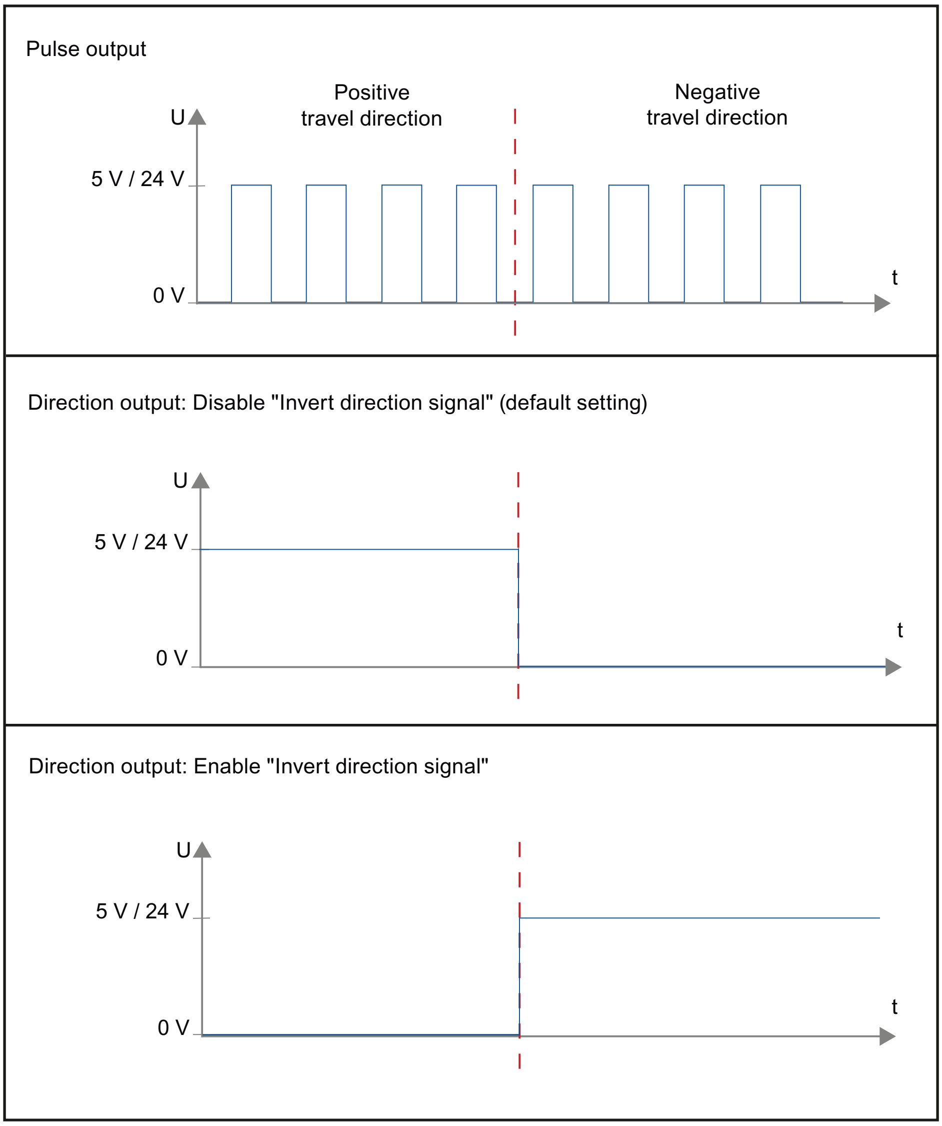
Homing (Page 2937)

Integration of the axis technology object (Page 2941)

Tools of the axis technology object (Page 2944)

Relationship between the travel direction and voltage level at the direction output

The direction output of the CPU specifies the travel direction of the drive. You configure the direction signal under "Mechanics" in the axis configuration. The relationships between configuration, direction output, and travel direction are presented in the following diagram:


Pulse output
Positive travel direction
Negative travel direction
U
5 V / 24 V
0 V
t
Direction output: Disable "Invert direction signal" (default setting)
U
5 V / 24 V
0 V
t
Direction output: Enable "Invert direction signal"
U
5 V / 24 V
0 V
t

If "Invert direction signal" is deactivated in the configuration, a level of 5 V / 24 V will be output at the direction output for a positive travel direction (the voltage depends on the hardware used). A 0 V level will be output at the direction output for positive travel direction if "Invert direction signal" is activated in the configuration.

See also

CPU outputs relevant for motion control (Page 2931)

How the pulse interface works (Page 2933)

Hardware and software limit switches (Page 2935)

Jerk limit (Page 2936)

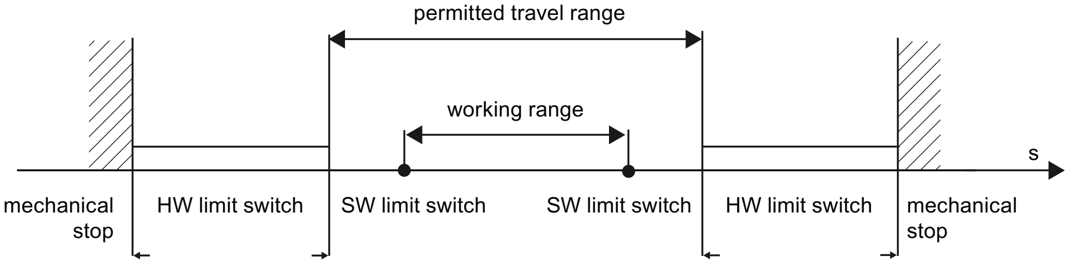
Homing (Page 2937)

Integration of the axis technology object (Page 2941)

Tools of the axis technology object (Page 2944)

Hardware and software limit switches

Use the hardware and software limit switches to limit the "permitted traversing range" and the "working range" of your axis technology object. The relationships are shown in the following diagram:



Hardware limit switches are limit switches that limit the maximum "permitted traversing range" of the axis. Hardware limit switches are physical switching elements that must be connected to interrupt-capable inputs of the CPU.

Software limit switches limit the "working range" of the axis. They should fall inside the hardware limit switches relative to the traversing range. Since the positions of the software limit switches can be flexibly set, the working range of the axis can be adapted on an individual basis, depending on the current traversing profile. In contrast to hardware limit switches, software limit switches are implemented exclusively via the software and do not require their own switching elements.

Hardware and software limit switches must be activated prior to use in the configuration or in the user program.. Software limit switches are only active after homing the axis.

See also

CPU outputs relevant for motion control (Page 2931)

How the pulse interface works (Page 2933)

Relationship between the travel direction and voltage level at the direction output (Page 2934)

Jerk limit (Page 2936)

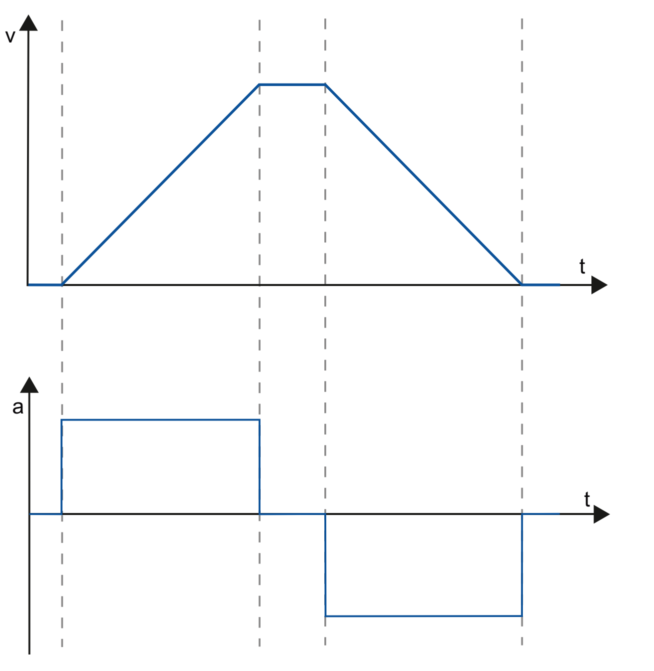
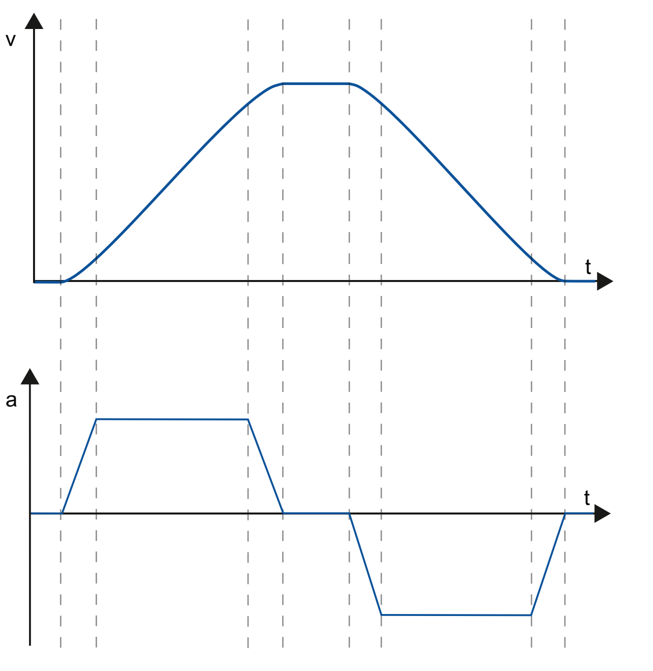
Homing (Page 2937)

Integration of the axis technology object (Page 2941)

Tools of the axis technology object (Page 2944)

Jerk limit

With the jerk limit you can reduce the stresses on your mechanics during an acceleration and deceleration ramp. The acceleration and deceleration value is not changed abruptly when the jerk limiter is active; it is gradually increased and decreased. The figure below shows the velocity and acceleration curve without and with jerk limit.

Travel without jerk limit	Travel with jerk limit
	

The jerk limit gives a "smoothed" velocity profile of the axis motion. This ensures soft starting and braking of a conveyor belt for example.

See also

Behavior of the axis when using the jerk limit (Page 2960)

CPU outputs relevant for motion control (Page 2931)

How the pulse interface works (Page 2933)

Relationship between the travel direction and voltage level at the direction output (Page 2934)

Hardware and software limit switches (Page 2935)

Homing (Page 2937)

Integration of the axis technology object (Page 2941)

Tools of the axis technology object (Page 2944)

Homing

Homing means matching the axis coordinates of the technology object to the real, physical location of the drive. For position-controlled axes the entries and displays for the position refer exactly to these axis coordinates. Therefore, agreement between the axis coordinates and the real situation is extremely important. This step is necessary to ensure that the absolute target position of the axis is also achieved exactly with the drive.

In the S7-1200 CPU, axis homing is implemented with the motion control instruction, "MC_Home". The following homing modes exist:

Homing modes

- Active homing

  In active homing mode, the motion control instruction "MC_Home" performs the required reference point approach. When the homing switch is detected, the axis is homed according to the configuration. Active traversing motions are aborted.

- Passive homing

  During passive homing, the motion control instruction "MC_Home" does not carry out any homing motion. The traversing motion required for this step must be implemented by the user via other motion control instructions. When the homing switch is detected, the axis is homed according to the configuration. Active traversing motions are not aborted upon start of passive homing.

- Direct homing absolute

  The axis position is set regardless of the homing switch. Active traversing motions are not aborted. The value of input parameter "Position" of motion control instruction "MC_Home" is set immediately as the reference point of the axis.

- Direct homing relative

  The axis position is set regardless of the homing switch. Active traversing motions are not aborted. The following statement applies to the axis position after homing:

  New axis position = current axis position + value of parameter "Position" of instruction "MC_Home".

See also

CPU outputs relevant for motion control (Page 2931)

How the pulse interface works (Page 2933)

Relationship between the travel direction and voltage level at the direction output (Page 2934)

Hardware and software limit switches (Page 2935)

Jerk limit (Page 2936)

Integration of the axis technology object (Page 2941)

Tools of the axis technology object (Page 2944)

Guidelines on use of motion control

The guidelines described here present the basic procedure for using motion control with the CPU S7-1200.

Requirements

To use the "Axis" technology object, you must create a project with a CPU S7-1200.

Procedure

Follow the steps below in the order given to use motion control with the CPU S7-1200. Use the following links for this purpose:

1. Add technological object Axis (Page 2945)
2. Working with the configuration dialog (Page 2947)
3. Download to CPU (Page 2987)
4. Function test of the axis in the commissioning window (Page 2988)
5. Programming (Page 2991)
6. Diagnostics of the axis control (Page 3010)

Overview of versions

The relationship between the relevant versions for S7-1200 Motion Control can be found in the following table:

Technology version

You can check the technology version currently selected in the "Add new object" dialog and in the Instructions > Technology task card. You select the technology version in the Instructions > Technology task card. If a technology object with an alternative version is added in the "Add new object" dialog, the technology version will also be changed.

Note

The selection of an alternative technology version will also affect the Motion Control instruction version (task card). The technology objects and Motion Control instructions will only be converted to the selected version upon compilation or "Load to device".

Version of the technology object

The version of a technology object can be checked in the inspector window under "Properties > General > Information" in the "Version" field.

To change the version, select the relevant version in the task card under Instructions > Technology and then the menu command Edit > Compile. If a technology object with an alternative version is added in the "Add new object" dialog, the technology object version will also be changed.

Deal with the causes of any errors if error information is displayed during compilation. Repeat compilation until it can be completed without errors.

Check the configuration of the technology objects once you have done so.

Motion Control instruction version

Proceed as follows to check the version of a Motion Control instruction:

1. Open the Program blocks > System blocks > Program resources folders in the navigator and select the required Motion Control instruction.
2. Select the Edit > Properties menu command.
3. You will find the Motion Control instruction version in the Version field of the Information tab.

If the Motion Control instruction version used is not in line with the following compatibility list, the relevant Motion Control instructions will be highlighted in the program editor.

Version compatibility list

	CPU Version V1.0	CPU Version V2.0	CPU Version V2.1
Technology version V1.0	Yes	Yes	Yes
Technology version V2.0	No	No	Yes

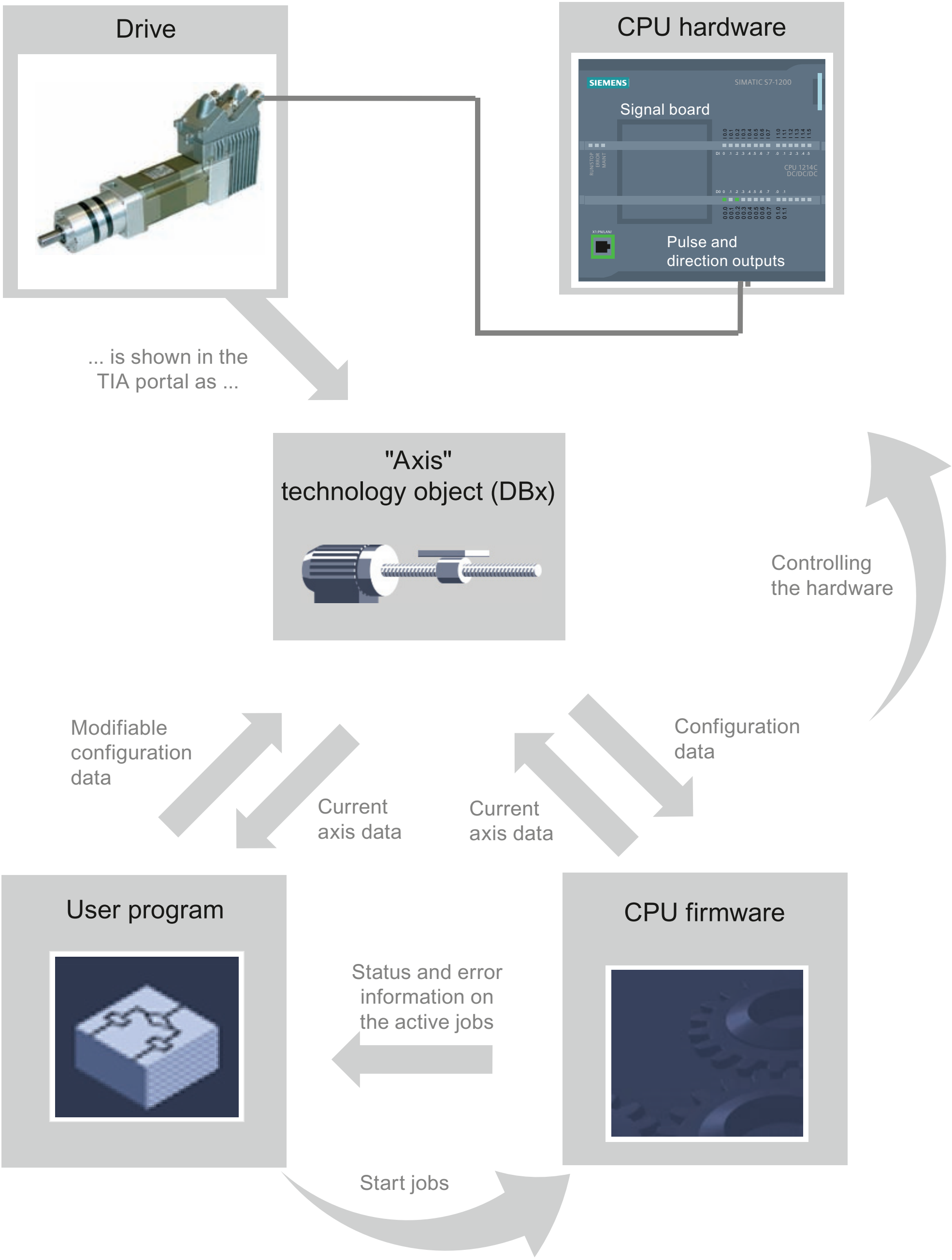
Technology object	Technology version V1.0	Technology version V2.0
Axis V1.0	Yes	No
Axis V2.0	No	Yes
Command table V2.0	No	Yes

Motion Control Instruction	Technology version V1.0	Technology version V2.0
MC_Power V1.0	Yes	No
MC_Power V2.0	No	Yes
MC_Reset V1.0	Yes	No
MC_Reset V2.0	No	Yes
MC_Home V1.0	Yes	No
MC_Home V2.0	No	Yes
MC_Halt V1.0	Yes	No
MC_Halt V2.0	No	Yes
MC_MoveAbsolute V1.0	Yes	No
MC_MoveAbsolute V2.0	No	Yes
MC_MoveRelative V1.0	Yes	No
MC_MoveRelative V2.0	No	Yes
MC_MoveVelocity V1.0	Yes	No
MC_MoveVelocity V2.0	No	Yes
MC_MoveJog V1.0	Yes	No
MC_MoveJog V2.0	No	Yes
MC_CommandTable V2.0	No	Yes
MC_ChangeDynamic V2.0	No	Yes

Techology object axis

Integration of the axis technology object

The following representation shows the relations between the hardware and software components which are implemented when using the "Axis" technology object:



CPU hardware

The physical drive is controlled and monitored by the CPU hardware.

Drive

The drive represents the unit of power unit and motor. Stepper motors or servo motors with a pulse interface may be used.

Technology object "Axis"

The physical drive including mechanics is mapped in the TIA Portal as an "axis" technology object. Configure the "Axis" technology object with the following parameters for this:

- Selection of the PTOs (Pulse Train Output) to be used and configuration of the drive interface
- Parameter for mechanics and gear transmission of the drive (or the machine or system)
- Parameter for position monitoring, for dynamic parameters and for homing

The configuration of the "Axis" technology object is saved in the technology object (data block). This data block also forms the interface between the user program and the CPU firmware. The current axis data is saved in the data block of the technology object at the runtime of the user program.

User program

You start motion control instructions jobs in the CPU firmware with the user program. The following jobs for controlling the axis are possible:

- Position axis absolutely
- Position axis relatively
- Move axis with velocity set point
- Run axis jobs as movement sequence (as of technology V2.0)
- Move axis in jog mode
- Stop axis
- Reference axis; set reference point
- Acknowledge error

You determine the command parameters with the input parameters of the Motion Control instructions and the axis configuration. The output parameters of the instruction give you up to date information about the status and any errors of the command.

Before starting a command for the axis, you must enable the axis with the motion control instruction "MC_Power".

You can read out configuration data and current axis data with the tags of the technology object. You can change single, changeable tags of the technology object (e.g. the current acceleration) from the user program.

CPU firmware

The motion control jobs started in the user program are processed in the CPU firmware. When using the axis control table, Motion Control jobs are triggered by operating the axis control table. The CPU firmware performs the following jobs depending on the configuration:

- Calculate the exact motion profile for motion jobs and emergency stop situations
- Control the drive enable and the pulse and direction signal
- Monitor the drive and the hardware and software limit switches
- Up to date feedback of status and error information to the motion control instructions in the user program
- Writing of current axis data into the data block of the technology object

See also

CPU outputs relevant for motion control (Page 2931)

Relationship between the travel direction and voltage level at the direction output (Page 2934)

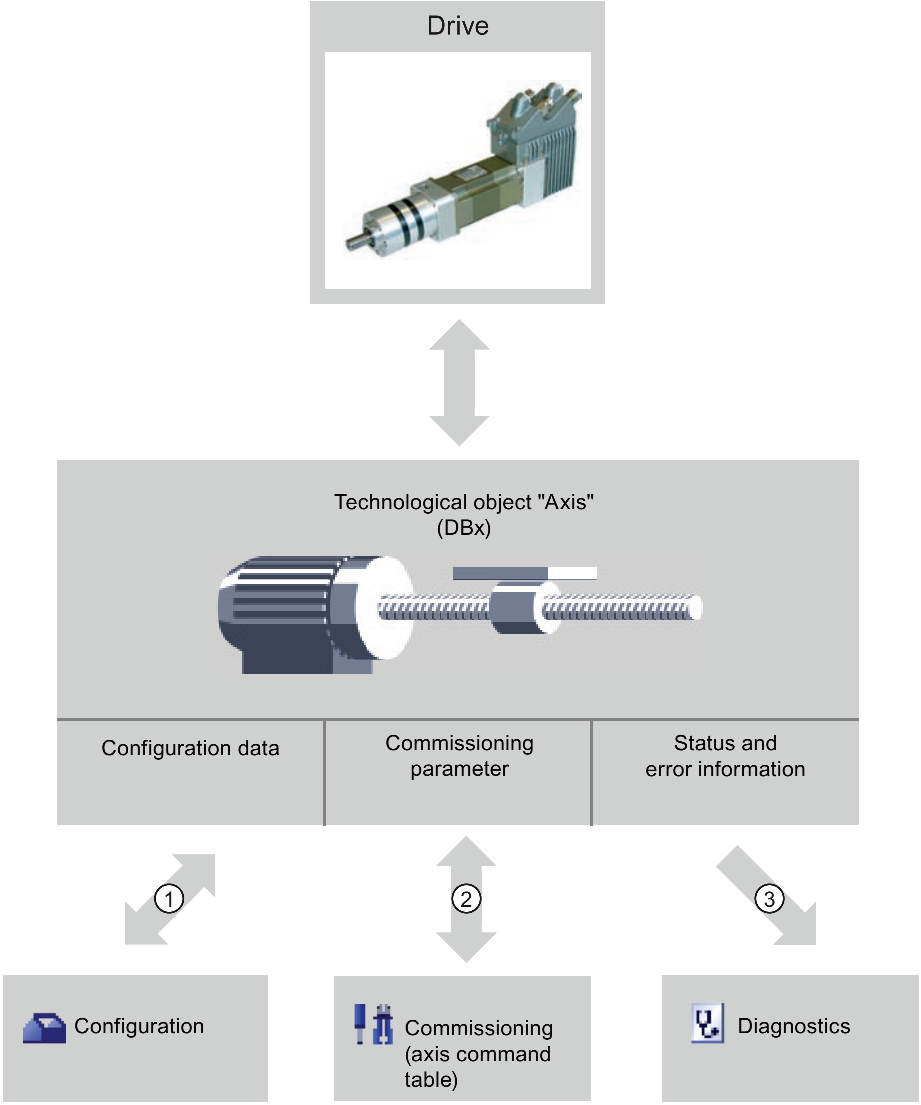
Tools of the axis technology object (Page 2944)

Hardware and software limit switches (Page 2935)

Homing (Page 2937)

Tools of the axis technology object

The TIA Portal provides the "Configuration", "Commissioning", and "Diagnostics" tools for the "Axis" technology object. The following representation shows the interaction of the three tools with the technology object and the drive:


Drive
Technological object "Axis"
(DBx)
Configuration data
Commissioning parameter
Status and error information
①
②
③
Configuration
Commissioning (axis command table)
Diagnostics

①	Reading and writing of configuration data of the technology object;
②	Drive control via the technology object; Reading axis status for display on the control panel
③	Readout of the current status and error information of the technology object.

Configuration

Use the "Configuration" tool to configure the following properties of the "Axis" technology object:

- Selection of the PTO to be used and configuration of the drive interface
- Properties of the mechanics and the transmission ratio of the drive (or machine or system)
- Properties for position monitoring, dynamics, and homing

Save the configuration in the data block of the technology object.

Commissioning

Use the "Commissioning" tool to test the function of your axis without having to create a user program. When the tool is started, the axis control table will be displayed. The following commands are available on the axis control table:

- Releasing and blocking the axis
- Move axis in jog mode
- Position axis in absolute and relative terms
- Home axis
- Acknowledge errors

The dynamic values can be adjusted accordingly for the motion commands. The axis control table also shows the current axis status.

Diagnostics

Use the "Diagnostics" tool to keep track of the current status and error information for the axis and drive.

See also

CPU outputs relevant for motion control (Page 2931)

Relationship between the travel direction and voltage level at the direction output (Page 2934)

Integration of the axis technology object (Page 2941)

Hardware and software limit switches (Page 2935)

Homing (Page 2937)

Commissioning the axis - Axis control panel (Page 2988)

Add technological object Axis

Proceed as follows to add an "Axis" technology object in the project tree:

Requirements

A project with a CPU S7-1200 has been created.

Procedure

1. Open the CPU folder in the project tree.
2. Open the technology objects folder.
3. Double-click "Add new object".

   The "Add new object" dialog opens.
4. Select the "Motion" technology.
5. Open the "Motion Control" folder.
6. Open the "S7-1200 Motion Control" folder.
7. Click on the version and select an alternative version of the technology if you want to add an axis from an older version.
8. Select the "TO_Axis_PTO" object.
9. Change the name of the axis in the "Name" input field to suit your needs.
10. Select the "Manual" option if you want to change the suggested data block number.
11. Click "More information" if you want to supplement user information for the technology object.
12. Click "OK" to add the technology object.

    Click "Cancel" to discard your entries.

Result

The new technology object is created and saved to the "Technology objects" folder in the project tree.

See also

Guidelines on use of motion control (Page 2938)

Configuring the axis technoloogy object

Working with the configuration dialog

You configure the properties of the technology object in the configuration window. Proceed as follows to open the configuration window of the technology object:

1. Open the group of the required technology object in the project tree.
2. Double-click the "Configuration" object.

The configuration is divided into the following categories:

- Basic parameters

  The basic parameters contain all the parameters which must be configured for a functioning axis.

- Extended parameters

  The advanced parameters include parameters to adapt to your drive or your plant.

Icons of the configuration window

Icons in the area navigation of the configuration show additional details about the status of the configuration:

✔	**The configuration contains default values and is complete.** The configuration contains only default values. With these default values you can use the technology object without additional changes.
✅	**The configuration contains values set by the user and is complete.** All input fields of the configuration contain valid values and at least one preset value has changed.
❌	**The configuration is incomplete or incorrect** At least one input field or drop-down list contains an invalid value. The corresponding field or the drop-down list is displayed on a red background. Click the roll-out error message to indicate the cause of error.
⚠	**The configuration is valid but contains warnings** Only one hardware limit switch is configured. Depending on the plant, the lacking configuration of a hardware limit switch may result in a hazard. The corresponding field or the drop-down list is displayed on a yellow background.

See also

Guidelines on use of motion control (Page 2938)

Basic parameters

Configuration - General

Configure the basic properties of the "Axis" technology object in the "General" configuration window.

Axis name:

Define the name of the axis or the name of the "Axis" technology object in this box. The technology object is listed under this name in the project navigation.

Hardware interface

The pulses are output to the power unit of the drive by fixed assigned digital outputs.

In CPUs with relay outputs, the pulse signal cannot be output on these outputs because the relays do not support the necessary switching frequencies. A signal board with digital outputs must be used to enable you to work with the PTO (Pulse Train Output) on these CPUs.

Note

The PTO requires the functionality of a fast counter (HSC) internally. The corresponding fast counter can therefore not be used elsewhere. The count can not be evaluated from its input address.

The assignment between PTO and HSC is fixed. When the user activates PTO1, it is connected to the HSC1. If the PTO2 is activated, this is connected with the HSC2.

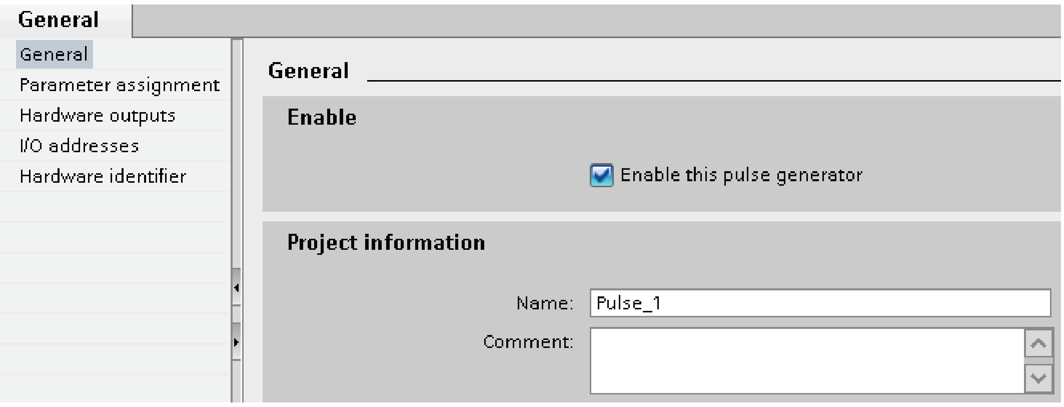
In the drop-down list "Pulse generator selection", select the PTO (Pulse Train Output) which are to provide the pulses for controlling the stepper motors or servo motors with a pulse interface. If the pulse generators and high-speed counters are not used elsewhere in the device configuration, the hardware interface can be configured automatically. In this case, the PTO selected in the drop-down list is displayed with a white background. The interfaces used will be listed in the "Output source", "Pulse output", "Direction output" and "Assigned fast counter" output fields.

Proceed as follows if you wish to change the interfaces or if the PTO could not be automatically configured (entry in the "Pulse generator selection" drop-down list is highlighted in red):

1. Click on the "Device configuration" button.

   The pulse generator device configuration opens.

   Enlarge the property window of the device configuration if the configuration of the pulse generator is not visible.

   ☑ Enable this pulse generator

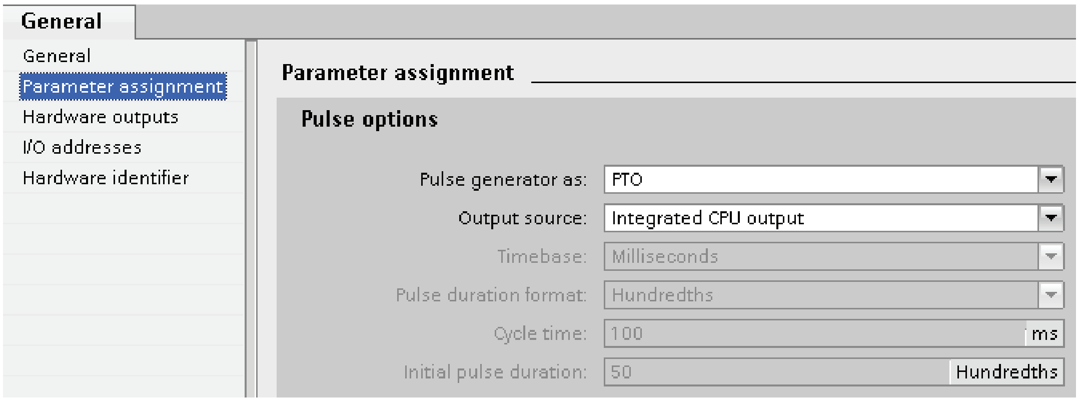
   Name: Pulse_1

   Comment:

2. Select the "Enable this pulse generator" check box.

3. Select the "Parameter assignment" entry in the block navigator.

   The "Parameter assignment" opens.



Parameter assignment

Pulse options

Pulse generator as: PTO

Output source: Integrated CPU output

Timebase: Milliseconds

Pulse duration format: Hundredths

Cycle time: 100 ms

Initial pulse duration: 50 Hundredths

4. In the "Pulse generator as:" dropdown list select the "PTO" entry.
5. In the "Output source:" dropdown list select the "Integrated CPU output" or "Signal board output" entry. The "Signal board output" entry can only be selected for PTO1 or for PTO1 and PTO2 depending on the plugged signal board. For more detailed information, see chapter: CPU outputs relevant for motion control (Page 2931)
6. Go back to the axis configuration.

   Unless the corresponding fast counter has already been used elsewhere, the PTO boxes of the "General" axis configuration are not shaded red. Correct the configuration based on the error messages if this is not the case.

User unit

Select the desired unit for the dimension system of the axis in the dropdown list. The selected unit is used for the further configuration of the "Axis" technology object and for the display of the current axis data.

The values at the input parameters (Position, Distance, Velocity, ...) of the Motion Control instructions also refer to this unit.

NOTICE
Later changing of the dimension system may not be converted correctly in all the configuration windows of the technology object. In this case check the configuration of all axis parameters. The values of the input parameters of the Motion Control instructions may have to be adapted to the new unit of measurement in the user program.

Extended parameters

Configuration - Drive interface

Configure the output for drive enable and the input for the "Drive ready" feedback signal of the drive in the "Drive signals" configuration window.

Drive enable is controlled by Motion Control instruction "MC_Power" and enables power to the drive. The signal is provided to the drive via the output to be configured.

The drive signals "Drive ready" to the CPU if it is ready to start executing travel after receipt of drive enable. The "Drive ready" signal is reported back to the CPU via the input to be configured.

If the drive does not have any interfaces of this type, you will not have to configure the parameters. In this case, select the value TRUE for the ready input.

See also

Configuration - Mechanics (Page 2950)

Configuration - Mechanics

Configure the mechanical properties of the drive in the "Mechanics" configuration window.

Increments per motor revolution

Configure the number of pulses required for one revolution of the motor in this field.

Limits (independent of the selected unit of measurement):

- 0 < Pulse per motor revolution ≤ 2147483647

Load distance per motor revolution

In this field, configure the load distance per motor revolution covered by the mechanical system of your unit.

Limits (independent of the selected unit of measurement):

- 0.0 < Load distance per motor revolution ≤ 1.0e12

Invert direction signal

You can adjust the direction output to the direction logic of the drive using the "Invert direction signal" check box.

- **Invert direction signal: deactivated**

  0 V level = negative travel direction

  5 V / 24 V level = positive travel direction (the actual voltage depends on the hardware used)

- **Invert direction signal: activated**

  0 V level = positive travel direction

  5 V / 24 V level = negative travel direction (the actual voltage depends on the hardware used)

See also

Configuration - Drive interface (Page 2950)

Relationship between the travel direction and voltage level at the direction output (Page 2934)

Position limits

Requirements for hardware limit switches

Use only hardware limit switches that remain permanently switched after being approached. This switching status may only be revoked after a return to the valid travel range.

See also

Configuration - Position limits (Page 2951)

Behavior of axis when position limits is tripped (Page 2953)

Changing the position limits configuration in the user program (Page 2954)

Configuration - Position limits

Configure the hardware and software limit switches of the axis in the "Position limits" configuration window.

Enable hardware limit switch

Activate the function of the low and high hardware limit switch with this check box. The hardware limit switches can be used for purposes of direction reversal during a reference point approach. For details, refer to the configuration description for homing.

Low / high HW limit switch input

Select the digital input for the low or high hardware limit switch from the drop-down list. The input must be interrupt-capable. The digital onboard CPU inputs and the digital inputs of a plugged signal board can be selected as inputs for the HW limit switches.

⚠ CAUTION
The digital inputs are set to a filter time of 6.4 ms by default. If these are used as hardware limit switches, undesired decelerations may occur. If this occurs, reduce the filter time for the relevant digital inputs. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

Active level

In the drop-down list, select the signal level available at the CPU when the hardware limit switch is approached.

- "Low level" selected

  0 V (FALSE) at CPU input corresponds to hardware limit switch approached
- "High level" selected

  5 V / 24 V (TRUE) at the CPU input = hardware limit switch approached (the actual voltage depends on the hardware used)

Enable software limit switch

Activate the function of the low and high software limit switch with this check box.

NOTICE
The enabled software limit switch only affects a homed axis.

High and low software limit switch

Enter the position value of the low and high software limit switch in these boxes.

Limits (independent of the selected unit of measurement):

- -1.0e12 ≤ low software limit switch ≤ 1.0e12
- -1.0e12 ≤ high software limit switch ≤ 1.0e12

The value of the high software limit switch must be greater than or equal to the value of the low software limit switch.

See also

Requirements for hardware limit switches (Page 2951)

Behavior of axis when position limits is tripped (Page 2953)

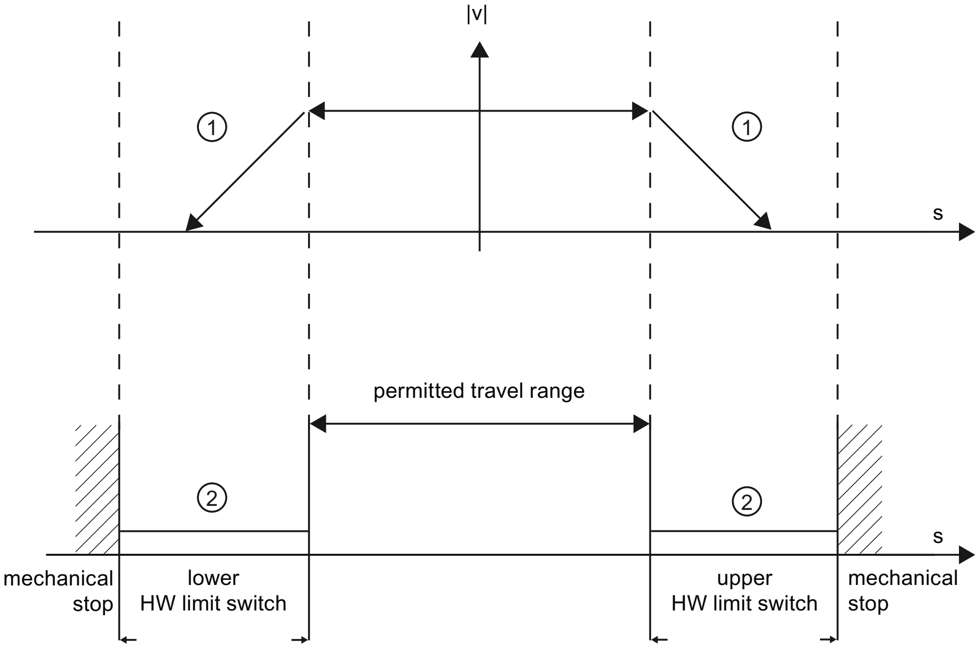
Changing the position limits configuration in the user program (Page 2954)

Configuration - Homing - Active (Page 2964)

Behavior of axis when position limits is tripped

Behavior of axis when hardware limit switches are approached

When the hardware limit switches are approached, the axis brakes to a standstill at the configured emergency stop deceleration. The specified emergency stop deceleration must be sufficient to reliably stop the axis before the mechanical stop. The following diagram presents the behavior of the axis after it approaches the hardware limit switches:



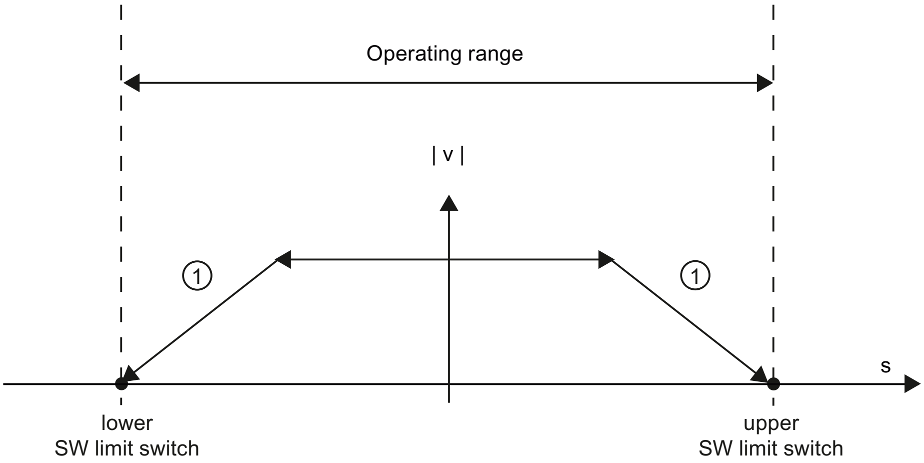
①	The axis brakes to a standstill at the configured emergency stop deceleration.
②	Range in which the HW limit switches signal the status "approached".

The "HW limit switch approached" error is displayed in the motion control instruction to be initiated, in "MC_Power", and in the technology object tags. Instructions for eliminating errors can be found in the Appendix under "List of ErrorIDs and ErrorInfos".

Behavior of axis when software limit switches are reached

If software limit switches are activated, an active motion is stopped at the position of the software limit switch. The axis is braked at the configured deceleration.

The following diagram presents the behavior of the axis until it reaches the software limit switches:



①	The axis brakes to a standstill at the configured deceleration.

The "SW limit switch reached" error is displayed in the motion control instruction to be initiated, in "MC_Power", and in the technology object tags. Instructions for eliminating errors can be found in the Appendix under "List of ErrorIDs and ErrorInfos".

The circumstances under which the "SW limit switch exceeded" error is displayed can be obtained in the topics "Software limit switches in conjunction with a homing operation (Page 3021)" and "Software limit switches in conjunction with dynamic changes (Page 3026)".

Use additional hardware limit switches if a mechanical endstop is located after the software limit switches and there is a risk of mechanical damage.

See also

Requirements for hardware limit switches (Page 2951)

Configuration - Position limits (Page 2951)

Changing the position limits configuration in the user program (Page 2954)

Changing the position limits configuration in the user program

You can change the following configuration parameters during user program runtime in the CPU:

Hardware limit switches

You can also activate and deactivate the hardware limit switches during runtime of the user program. Use the following technology object tag for this purpose:

- <Axis name>.Config.PositionLimits_HW.Active

Please refer to the description of the technology object tags in the Appendix for information on when changes to the configuration parameter become effective.

Software limit switches

You can also activate and deactivate the software limit switches and change their position values during runtime of the user program. Use the following technology object tags for this purpose:

- <Axis name>.Config.PositionLimits_SW.Active

  for activating and deactivating the software limit switches
- <Axis name>.Config.PositionLimits_SW.MinPosition

  for changing the position of the low software limit switch
- <Axis name>.Config.PositionLimits_SW.MaxPosition

  for changing the position of the high software limit switch

Refer to the description of technology object tags in the Appendix for information on when changes to the configuration parameters take effect.

See also

MC_ChangeDynamic: Changing the dynamic settings for the axis (as of technology object "Axis" V2.0) (Page 1846)

Requirements for hardware limit switches (Page 2951)

Configuration - Position limits (Page 2951)

Behavior of axis when position limits is tripped (Page 2953)

Dynamics

Configuration - General dynamics

Configure the maximum velocity, the start/stop velocity, the acceleration and deceleration and the jerk limit (as of technology object "Axis" V2.0) in the "General dynamics" configuration window.

Velocity limiting unit

Select the unit of measurement with which you want to set the velocity limits in the dropdown list. The unit set here depends on the unit of measurement set under "Configuration - General" and serves only for easier input.

Maximum velocity / Start/stop velocity

Define the maximum permissible velocity and the start/stop velocity of the axis in these boxes. The start/stop velocity is the minimum permissible velocity of the axis.

Limit values:

The limits indicated below refer to the "Pulses/s" unit of measurement:

- Technology object Axis V2.0
  - 2 ≤ start/stop velocity ≤ 20000 (signal board 20kHz)

    2 ≤ start/stop velocity ≤ 200000 (signal board 200kHz)

    2 ≤ start/stop velocity ≤ 100000 (on-board CPU outputs)
  - 2 ≤ maximum velocity ≤ 20000 (signal board 20kHz)

    2 ≤ maximum velocity ≤ 200000 (signal board 200kHz)

    2 ≤ maximum velocity ≤ 100000 (on-board CPU outputs)
- Technology object Axis V1.0
  - 2 ≤ start/stop velocity ≤ 20000 (signal board 20kHz)

    2 ≤ start/stop velocity ≤ 100000 (signal board 200kHz)

    2 ≤ start/stop velocity ≤ 100000 (on-board CPU outputs)
  - 2 ≤ maximum velocity ≤ 20000 (signal board 20kHz)

    2 ≤ maximum velocity ≤ 100000 (signal board 200kHz)

    2 ≤ maximum velocity ≤ 100000 (on-board CPU outputs)

The value of the maximum velocity must be greater or equal to the value of the start/stop velocity.

The limit values for other units of measurement must be converted by the user to conform to the given mechanics.

Acceleration / Delay - Ramp-up time / Ramp-down time

Set the desired acceleration in the "Ramp-up time" or "Acceleration" boxes. The desired deceleration can be set in the "Deceleration time" or "Deceleration" boxes.

The relation between the ramp-up time and acceleration and the deceleration time and deceleration is shown in the following equations:

$$\textit{Rampup time} = \frac{\textit{Maximum velocity} - \textit{Start/stop velocity}}{\textit{Acceleration}}$$

$$\textit{Deceleration time} = \frac{\textit{Maximum velocity} - \textit{Start/stop velocity}}{\textit{Deceleration}}$$

Motion jobs started in the user program are performed with the selected acceleration / deceleration.

Limit values:

The limits indicated below refer to the "Pulses/s^2" units of measurement:

- 0.28 ≤ acceleration ≤ 9.5e9
- 0.28 ≤ deceleration ≤ 9.5e9

The limits for other units of measurement must be converted to conform to the given mechanics.

Note

Changes to the velocity limits ("start/stop velocity" and "maximum velocity") influence the acceleration and deceleration values of the axis. The ramp-up and deceleration times are retained.

Activate jerk limit (as of technology object Axis V2.0)

Activate the jerk limit with this check box.

Note

If an error occurs, the axis decelerates with the configured emergency stop deceleration. An activated jerk limit is not considered here.

Smoothing time/jerk (as of technology object "Axis" V2.0)

You can input the parameters of the jerk limit in the "Smoothing time" field or alternatively in the "Jerk" field.

- Set the desired jerk for acceleration and deceleration ramp in the "Jerk" field.
- Set the desired smoothing time for the acceleration ramp in the "Rounding time" field.

Note

The set smoothing time visible in the configuration only applies to the acceleration ramp.

If the values for acceleration and deceleration differ, the smoothing time of the deceleration ramp is calculated according to the jerk of the acceleration ramp and used. (See also Behavior of the axis when using the jerk limit (Page 2960)

The smoothing time of the deceleration is adapted as follows:

- Acceleration > deceleration

  The smoothing time used for the deceleration ramp is shorter than that for the acceleration ramp.
- Acceleration < deceleration

  The smoothing time used for the deceleration ramp is shorter than that for the acceleration ramp.
- Acceleration = deceleration

  The smoothing times of the acceleration and deceleration ramp are equal.

The relation between smoothing times and jerk is shown in the following equation:

$$\text{Rounding off time (acceleration ramp)} = \frac{\text{Acceleration}}{\text{Step}}$$

$$\text{Rounding off time (deceleration ramp)} = \frac{\text{Deceleration}}{\text{Step}}$$

Motion jobs started in the user program are performed with the selected jerk.

Limit values:

The limits indicated below refer to the Pulses/s^3 units of measurement:

- $0.04 \leq \text{jerk} \leq 1.5\text{e}8$

The limits for other units of measurement must be converted to conform to the given mechanics.

See also

Behavior of the axis when using the jerk limit (Page 2960)

Configuration - Dynamics emergency stop (Page 2959)

Changing the configuration of dynamics in the user program (Page 2961)

Configuration - Dynamics emergency stop

Configure the emergency stop deceleration of the axis in the "Dynamics emergency stop" configuration window. When an error occurs and when the axis is disabled with motion control instruction "MC_Power" (input parameter StopMode = 0), the axis is brought to a standstill with this deceleration.

Velocity limits

The velocity values configured in the "General dynamics" configuration window are once again displayed in this information area.

Deceleration

Set the deceleration value for emergency stop in the "Emergency stop deceleration" or "Emergency stop ramp-down time" field.

The relation between emergency stop deceleration time and emergency stop deceleration is shown in the following equation:

$$\textit{Emergency stop deceleration time} = \frac{\textit{Maximum velocity} - \textit{Start/stop velocity}}{\textit{Emergency stop deceleration}}$$

The specified emergency stop deceleration must be sufficient to bring the axis to a standstill in a timely manner in the event of an emergency (for example, when the hardware limit switch is approached prior to reaching the mechanical endstop).

The configured maximum velocity of the axis must be used as a basis for selecting the emergency stop deceleration.

Limit values:

The limits indicated below refer to the "Pulses/s^2" units of measurement:

- 0.28 ≤ emergency stop deceleration ≤ 9.5e9

The limits for other units of measurement must be converted to conform to the given mechanics.

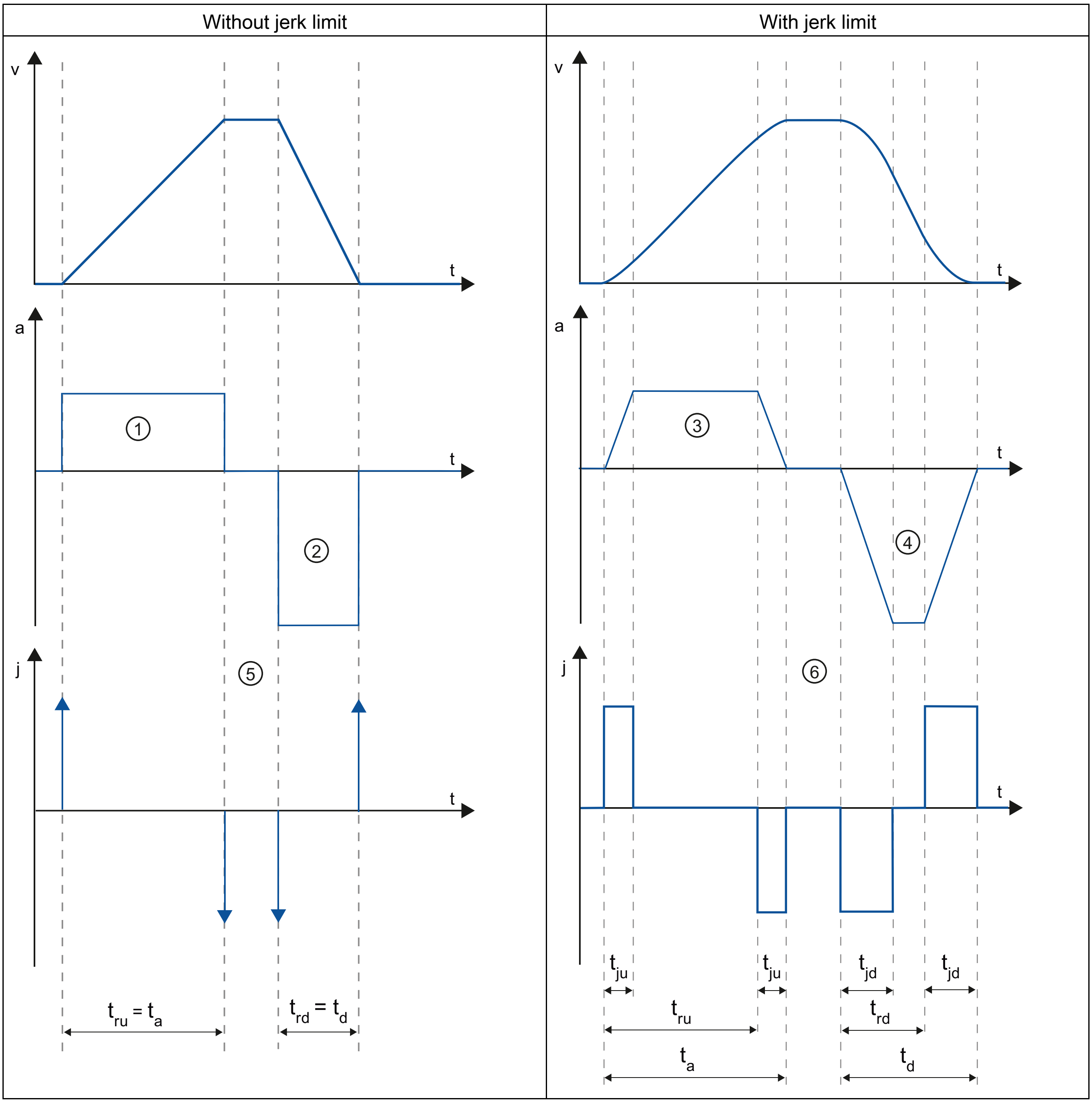
See also

Configuration - General dynamics (Page 2955)

Changing the configuration of dynamics in the user program (Page 2961)

Behavior of the axis when using the jerk limit

Axis acceleration and deceleration is not stopped abruptly when the jerk limit is activated; it is adjusted gently according to the set step or rounding off time. The diagram below details the behavior of the axis with and without activated jerk limit:



t	Time axis
v	Velocity
a	Acceleration
j	Step
t_{ru}	Rampup time
t_a	Time taken for the axis to accelerate
t_{rd}	Deceleration time
t_d	Time taken for the axis to decelerate
t_{ju}	Smoothing time of the acceleration ramp
t_{jd}	Smoothing time of the deceleration ramp

The example shows travel in which the deceleration value ② is twice the acceleration value ①. The resulting ramp-down time t_{rd} is therefore only half the length of the ramp-up time t_{ru}.

Acceleration ① and deceleration ② change abruptly without a jerk limit. Acceleration ① and deceleration ② change gradually with activated jerk limiter. As the jerk applies to entire motion, the rate is the same for the increase and decrease in acceleration and deceleration.

The step value j becomes infinitely high ⑤ as soon as the change is made without jerk limit. The step is limited to the configured value ⑥ when the jerk limit is activated.

The smoothing time t_{ju} given in the configuration applies to the acceleration ramp. The deceleration ramp smoothing time t_{ju} is calculated using the configured jerk value and the configured deceleration.

See also

Configuration - General dynamics (Page 2955)

Changing the configuration of dynamics in the user program

You can change the following configuration parameters during user program runtime in the CPU:

Acceleration and deceleration

You can also change the values for acceleration and deceleration during runtime of the user program. Use the following technology object tags for this purpose:

- <Axis name>.Config.DynamicDefaults.Acceleration

  for changing acceleration
- <Axis name>.Config.DynamicDefaults.Deceleration

  for changing deceleration

Refer to the description of technology object tags in the Appendix for information on when changes to the configuration parameters take effect.

Emergency stop deceleration

You can also change the value for the emergency stop deceleration during runtime of the user program. Use the following technology object tag for this purpose:

- <Axis name>.Config.DynamicDefaults.EmergencyDeceleration

Please refer to the description of the technology object tags in the Appendix for information on when changes to the configuration parameter become effective.

⚠ WARNING
After changes to this parameter, it may be necessary to adapt the positions of the hardware limit switches and other safety-relevant settings.

Jerk limit (as of technology object "Axis" V2.0)

You can also activate and deactivate the jerk limit at runtime of the user program and change the value for the jerk. Use the following technology object tag for this purpose:

- <Axis name>.Config.DynamicDefaults.JerkActive

  for activating and deactivating the jerk limit
- <Axis name>.Config.DynamicDefaults.Jerk

  for changing the jerk

Please refer to the description of the technology object tags in the Appendix for information on when changes to the configuration parameter become effective.

See also

MC_ChangeDynamic: Changing the dynamic settings for the axis (as of technology object "Axis" V2.0) (Page 1846)

Configuration - General dynamics (Page 2955)

Configuration - Dynamics emergency stop (Page 2959)

Homing (technology object "Axis" as of V2.0)

Configuration - Homing - General

Configure the reference point switch input for active and passive homing in the "Homing - General" configuration window.

Reference point switch input

Select the digital input for the reference point switch from the drop-down list box. The input must be able to generate an interrupt. The onboard CPU inputs and inputs of an inserted signal board can be selected as inputs for the reference point switch.

Note

The digital inputs are set to a filter time of 6.4 ms by default.

When the digital inputs are used as a reference point switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the reference point switch, the reference point may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

The specified filter time must be less than the duration of the input signal at the reference point switch.

See also

Sequence - Active homing (Page 2967)

Configuration - Homing - Passive

Configure the necessary parameters for passive homing in the "Homing - Passive" configuration window.

The movement for passive homing must be triggered by the user (e.g. using an axis motion command). Passive homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 2.

Side of the reference point switch

This is where you select whether the axis is to be homed on the low or high side of the reference point switch.

Reference point position

The position for which parameters were assigned in the Motion Control instruction "MC_Home" is used as the reference point position.

Note

If passive homing is carried out without an axis motion command (axis at a standstill), homing will be executed upon the next rising or falling edge at the reference point switch.

Configuration - Homing - Active

Configure the necessary parameters for active homing in the "Active homing" configuration window. Active homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 3.

Permit auto reverse at the hardware limit switch

Activate the check box to use the hardware limit switch as a reversing cam for the reference point approach. The hardware limit switches must be enabled for the reversal of direction (at least the hardware limit switch in the direction of approach must be configured).

If the hardware limit switch is reached during active homing, the axis brakes at the configured deceleration (not with the emergency stop deceleration) and reverses direction. The homing switch is then sensed in reverse direction.

If the direction reversal is not active and the axis reaches the hardware limit switch during active homing, the reference point approach is aborted with an error and the axis is braked at the emergency stop deceleration.

NOTICE
If possible, use one of the following measures to ensure that the machine does not travel to a mechanical endstop in the event of a direction reversal: • Keep the approach velocity low • Increase the configured acceleration/deceleration • Increase the distance between hardware limit switch and mechanical stop

Approach/homing direction

With the direction selection, you determine the approach direction used during active homing to search for the homing switch, as well as the homing direction. The homing direction specifies the travel direction the axis uses to approach the configured side of the homing switch to carry out the homing operation.

Side of the homing switch

This is where you select whether the axis is to be homed on the low or high side of the homing switch.

Velocity

In this field, specify the velocity at which the homing switch is to be searched for during the reference point approach.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ approach velocity ≤ maximum velocity

Homing velocity

Specify in this field the velocity at which the homing switch is to be approached for homing.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ Homing velocity ≤ Maximum velocity

Home position offset

If the desired reference position deviates from the position of the homing switch, the home position offset can be specified in this field.

If the value does not equal 0, the axis executes the following actions following homing at the homing switch:

1. Move the axis at the homing velocity by the value of the home position offset
2. Upon reaching the "Home position offset", the axis is at the home position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

Limits (independent of the selected unit of measurement):

- -1.0e12 ≤ home position offset ≤ 1.0e12

Home position

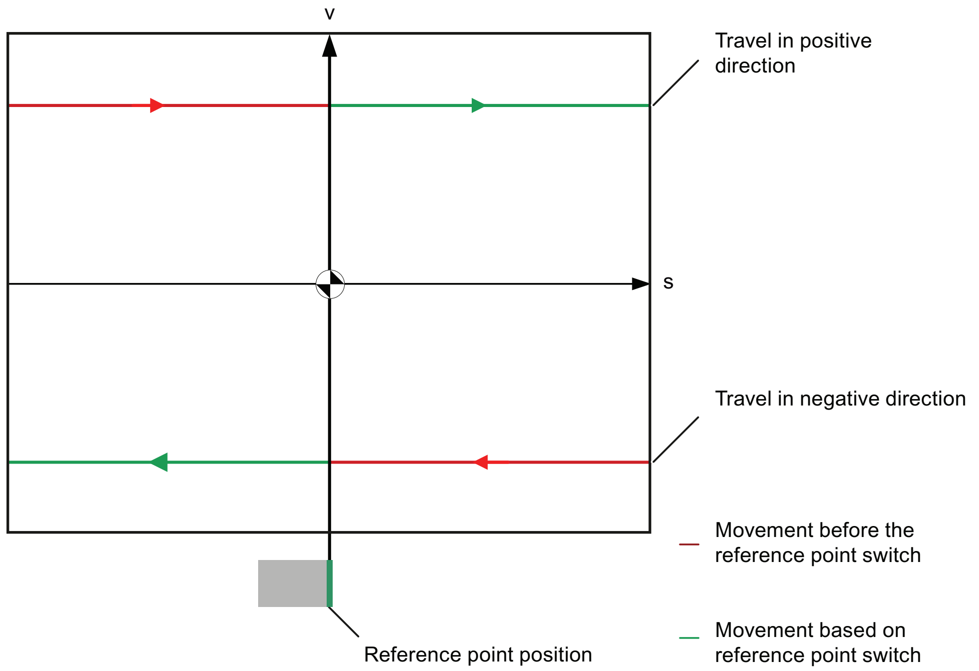
The position for which parameters were assigned in the Motion Control instruction "MC_Home" is used as the reference point position.

Sequence - Passive homing

Passive homing is started with Motion Control instruction "MC_Home" (input parameter Mode = 2). Input parameter "Position" specifies the absolute reference point position.

The diagram below shows an example of a characteristic curve for passive homing with the following configuration parameters:

- "Reference point switch side" = "High side"


v
s
Travel in positive direction
Travel in negative direction
Movement before the reference point switch
Movement based on reference point switch
Reference point position

Movement towards reference point switch (red section of curve)

The Motion Control instruction "MC_Home" does not itself carry out any homing motion when passive homing is started. The travel required for reaching the reference point switch must be implemented by the user via other motion control instructions such as "MC_MoveRelative". The tag <axis name>.StatusBits.HomingDone remains TRUE during passive homing if the axis has already been homed.

Axis homing (transition from red to green section of curve)

The axis is homed when the configured side of the reference point switch is reached. The current position of the axis is set to the reference point position. This is specified at the "Position" parameter of the "MC_Home" Motion Control instruction. The variable <axis name>.StatusBits.HomingDone will be set to "TRUE" if the axis has not been been homed before. The travel previously started is not cancelled.

Movement beyond reference point switch (green section of curve)

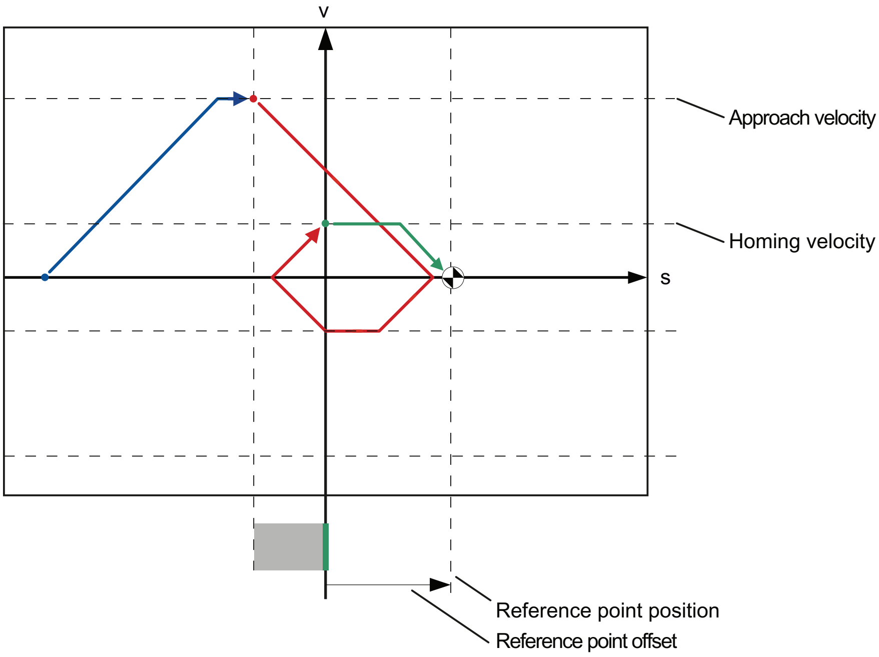
Following homing at the reference point switch, the axis continues and completes the previously started travel with the corrected axis position.

Sequence - Active homing

You start active homing with motion control instruction "MC_Home" (input parameter Mode = 3). The "Position" input parameter specifies the absolute home position. Alternatively, you can start active homing on the axis command table for test purposes.

The diagram below shows an example of a characteristic curve for an active reference point approach with the following configuration parameters:

- "Approach/homing direction" = "Positive direction"
- "Side of the homing switch" = "Top side"
- Value of "home position offset" > 0



Search for homing switch (blue curve section)

When active homing starts, the axis accelerates to the configured "approach velocity" and searches at this velocity for the homing switch. The tag <axis name>.StatusBits.HomingDone is set to FALSE.

Reference point approach (red curve section)

When the homing switch is detected, the axis in this example brakes and reverses, to be homed to the configured side of the homing switch at the configured homing velocity. Homing causes the tag <axis name>.StatusBits.HomingDone to change to TRUE.

Travel to home position offset (green curve segment)

After homing, the axis moves at the homing velocity along the path to the home position offset. There the axis is at the homing point position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

See also

Configuration - Homing - General (Page 2962)

Changing the homing configuration in the user program

You can change the following configuration parameters in the CPU during user program runtime as of technology object "Axis" V2.0:

Passive homing

You can change the side of the homing switch for passive homing during the user program runtime. Use the following technology object tag for this purpose:

- <Axis name>.Config.Homing.SidePassiveHoming

  for changing the side of the homing switch

Please refer to the description of the technology object tags in the Appendix for information on when changes to the configuration parameter become effective.

Active homing

You can change the direction of approach, the side of the homing switch, the approach velocity, the homing velocity, and the home position offset for active homing during the program runtime of the user program. Use the following technology object tags for this purpose:

- <Axis name>.Config.Homing.AutoReversal

  for changing "auto reverse at the HW limit switch"
- <Axis name>.Config.Homing.Direction

  for changing "approach / homing direction"
- <Axis name>.Config.Homing.SideActiveHoming

  for changing the "side of the homing switch"
- <Axis name>.Config.Homing.FastVelocity

  for changing the "velocity"
- <Axis name>.Config.Homing.SlowVelocity

  for changing the "homing velocity"
- <Axis name>.Config.Homing.Offset

  for changing "home position offset"

Please refer to the description of the technology object tags in the Appendix for information on when changes to the configuration parameter become effective.

See also

MC_ChangeDynamic: Changing the dynamic settings for the axis (as of technology object "Axis" V2.0) (Page 1846)

Technology object command table

Use of the command table technology object

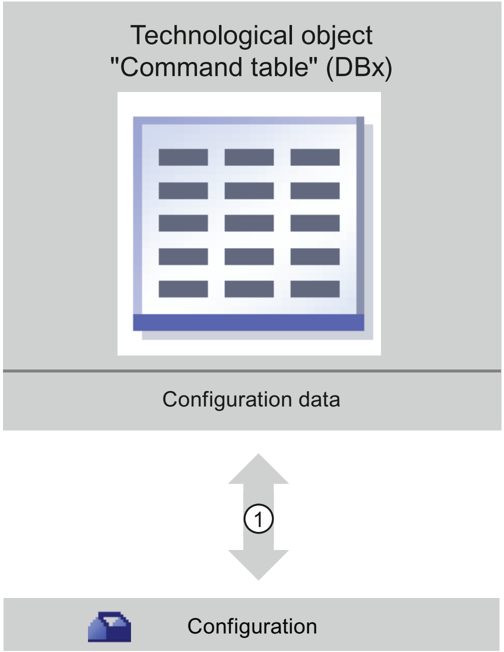
The Motion Control instruction "Command table" allows you to combine multiple individual axis control jobs in one movement sequence. The technology object can be used for technology as of Version V2.0.

You configure the movement sequence as a table in a configuration dialog.

The motion profile of the movement sequences can be checked on a graph before the project is loaded to the CPU. The command tables created are then linked to an axis and used in the user program with the "MC_CommandTable" Motion Control instruction. You can process part or all of the command table.

Command table technology object tools

The "Configuration" tool is provided in the TIA Portal for the "Command Table" technology object. The representation below shows the interaction of the tool with the technology object:



①	Writing and reading the configuration of the technology object

Configuration

Configure the following properties of the "Command Table" technology object with the "Configuration" tool:

- You can create one or more movement sequences by configuring individual jobs.
- You can configure the graphic display to check your movement sequence using an axis already configured or a configurable default axis.

The movement sequence data are saved in the data block of the technology object.

Adding the technological object command table

Proceed as follows to add a "Command table" technology object in the project tree:

Prerequisites

- A project with a CPU S7-1200 has been created.
- The CPU firmware version is V2.1 or higher

Procedure

1. Open the CPU folder in the project tree.
2. Open the technology objects folder.
3. Double-click "Add new object".

   The "Add new object" dialog opens.
4. Select the "Motion" technology.
5. Open the "Motion Control" folder.
6. Open the "S7-1200 Motion Control" folder
7. Select the version "V2.0" of the "S7-1200 Motion Control" folder (click on the entry for the version).
8. Select the "TO_CommandTable" object.
9. Change the name of the command table in the "Name" input field to suit your needs.
10. Select the "Manual" option if you want to change the suggested data block number.
11. Click "More information" if you want to supplement user information for the technology object.
12. Click "OK" to add the technology object.

    Click "Cancel" to discard your entries.

Result

The new technology object is created and saved to the "Technology objects" folder in the project tree.

Configuring the command table technology object

Working with the configuration dialog

You configure the properties of the technology object in the configuration window. Proceed as follows to open the configuration window of the technology object:

1. Open the group of the required technology object in the project tree.
2. Double-click the "Configuration" object.

The configuration is divided into the following categories:

- Basic parameters

  The basic parameters contain all parameters which must be configured for a functional command table.

- Extended parameters

  The extended parameters contain the parameters of the default axis or display the parameter values of the axis selected.

Icons of the configuration window

Icons in the area navigation of the configuration show additional details about the status of the configuration:

✓	The configuration contains default values and is complete. The configuration contains only default values. With these default values, you can use the technology object without additional changes.
✓	The configuration contains values set by the user and is complete. All input fields of the configuration contain valid values and at least one preset value has changed.
✖	The configuration is incomplete or incorrect At least one input field or drop-down list contains an invalid value. The corresponding field or the drop-down list is displayed on a red background. Click the roll-out error message to display the cause of the error.
⚠	The configuration contains mutually incompatible parameter values The configuration contains parameter values that contradict each other either in size or logic. The corresponding field or the drop-down list is displayed on a yellow background.

See also

Guidelines on use of motion control (Page 2938)

Basic parameters

Configuration - General

Configure the name of the technology object in the "General" configuration window.

Name

Define the name of the command table or the name of the "Command table" technology object in this field. The technology object will be listed under this name in the project tree.

See also

Configuration - Command table (Page 2972)

Shortcut menu commands - Command table (Page 2975)

Working with the trend diagram (Page 2977)

Shortcut menu commands - Curve chart (Page 2981)

Transition from "Complete command" to "Blend motion" (Page 2982)

Changing the command table configuration in the user program (Page 2983)

Configuration - Command table

Create the desired movement sequence in the "Command Table" configuration window and check the result against the graphic view in the trend diagram.

Note

Small deviations are possible between the time behavior and position in the trend shown and the real movement of the axis. Movements in response to software limit switches being reached are not shown.

Activate warnings

Activate the display of warnings in the command table with this checkbox.

Use axis parameters of

From the drop-down list, select which axis parameters are to be used for selecting the graphic view of and checking the movement sequence. Select "Default axis" if you have yet to add an axis to the "Technology object" folder or wish to use value which have not been configured in any of the available axes. You configure the properties of the default axis under "Advanced parameters".

The axis parameters of the axis selected at the "Axis" parameter will be used to process the command table in the user program.

Column: Step

Shows the step number of the command.

Column: Command type

In this column, select the command types which are to be used for processing the command table. Up to 32 jobs can be entered. The commands will be processed in sequence. You can choose between the following entries and command types:

- Empty

  The entry serves as a placeholder for any commands to be added. The empty entry is ignored when the command table is processed.
- Halt

  Pause axis
  (the command only takes effect after a "Velocity setpoint" command)
- Positioning Relative

  Position axis relatively
- Positioning Absolute

  Position axis absolutely
- Velocity setpoint

  Move axis at set velocity
- Wait

  Waits until the given period is over. Wait does not stop an active traversing motion.
- Separator

  Adds a Separator line above the selected line. The Separator line acts as a range limit for the graphic display of the trend view.

  Use the Separator lines if you wish to process parts of the command table.

Column: Position

Enter the position or travel path for the selected command in this column:

- Command "Positioning relative"

  The command will move the axis by the the given travel path.
- Command "Positioning absolute"

  The command will move the axis by the the given position.
- Separator

  The value given specifies the start position for the graphic display.

Limit values (independent of the selected user unit):

- -1.0e12 ≤ position / distance ≤ -1.0e-12
- 1.0e-12 ≤ position / distance ≤ 1.0e12
- Position / travel path = 0.0

Column: Velocity

In this column, you enter the velocity for the selected command:

- **Command "Positioning relative"**

  The command will move the axis at the the given velocity.

  The given velocity will not be reached if the travel path selected is not large enough.
- **Command "Positioning absolute"**

  The command will move the axis at the the given velocity.

  The given velocity will not be reached if the target position is too close to the starting position.
- **Command " Velocity setpoint"**

  The command will move the axis at the the given velocity.

  The given velocity will not be reached during the command if too short a runtime is selected.

Limit values (independent of the selected user unit):

- For the jobs: "Positioning relative" and "Positioning absolute"
  - 1.0e-12 ≤ velocity ≤ 1.0e12
- For the command: "Velocity setpoint"
  - -1.0e12 ≤ velocity ≤ -1.0e-12
  - 1.0e-12 ≤ velocity ≤ 1.0e12
  - Velocity = 0.0

Column: Duration

Enter the duration of the selected command in this column:

- **Command " Velocity setpoint"**

  The command will move the axis for the specified duration. The duration includes both the acceleration phase and the constant travel phase. The next command will be processed once the duration is over.
- **Command "Wait"**

  Waits until the given duration is over.

Limit values (independent of the selected user unit):

- 0.001s ≤ duration ≤ 64800s

Column: Next step

Select the mode of transition to the next step from the drop-down list:

- Complete command

  The command will be completed. The next command will be processed immediately.

- Blend motion

  The motion of the current command will be blended with the motion of the following command. The transition mode "Blend motion" is available with command types "Positioning Relative" and "Positioning Absolute".

  Motion will be blended with motions of the following command types:

  – Positioning Relative

  – Positioning Absolute

  – Velocity setpoint

No blending occurs with other command types.

For the exact behavior of the axis when a command is appended or overlapped, see: Transition from "Complete command" to "Blend motion" (Page 2982)

Column: Step code

Enter a numerical value / bit pattern in this column which is to be output at the "StepCode" output parameter of the "MC_CommandTable" Motion Control instruction while the command is being processed.

Limit values:

- 0 ≤ code number ≤ 65535

See also

Configuration - General (Page 2971)

Shortcut menu commands - Command table (Page 2975)

Working with the trend diagram (Page 2977)

Shortcut menu commands - Curve chart (Page 2981)

Transition from "Complete command" to "Blend motion" (Page 2982)

Changing the command table configuration in the user program (Page 2983)

Shortcut menu commands - Command table

The following shortcut menu commands are available in the command table:

Insert empty line

Adds an empty line above the selected line.

This shortcut menu command can only be executed if there are enough empty lines at the end of the command table.

Add empty line

Adds an empty line below the selected line.

This shortcut menu command can only be executed if there are enough empty lines at the end of the command table.

Insert separator line

Adds a separator line above the selected line.

You cannot have two consecutive separator lines.

Add separator line

Adds a separator line below the selected line.

You cannot have two consecutive separator lines, nor can you add a separator line at the end of the command table.

Cut

Removes the selected lines or content of the selected cell and saves them/it in the clipboard.

Selected lines will be deleted and the subsequent lines of the command table shifted up.

Copy

Copies the selected lines or content of the selected cell and saves them/it in the clipboard.

Paste

- Selected lines:

  Pastes the lines from the clipboard into the table above the selected line.
- Selected cell:

  Pastes the content of the clipboard into the selected line.

This shortcut menu command can only be executed if there are enough empty lines at the end of the command table.

Replace

Replaces the selected lines with the lines in the clipboard.

Delete

Deletes the selected lines. The lines below in the command table shift up.

See also

Configuration - General (Page 2971)

Configuration - Command table (Page 2972)

Working with the trend diagram (Page 2977)

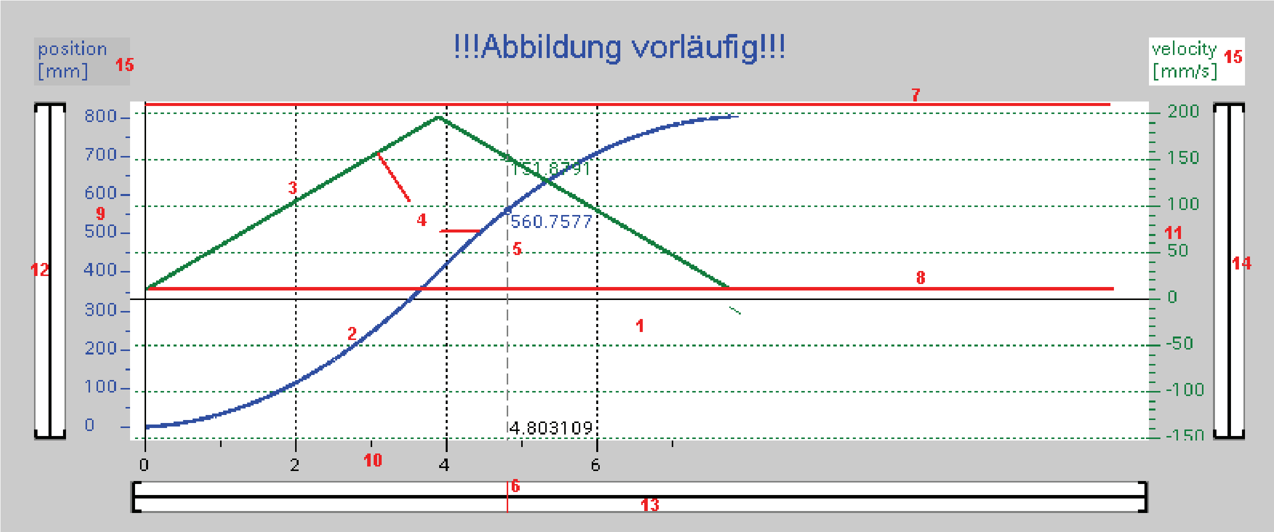
Shortcut menu commands - Curve chart (Page 2981)

Transition from "Complete command" to "Blend motion" (Page 2982)

Changing the command table configuration in the user program (Page 2983)

Working with the trend diagram

Trend view and components



①	Trend view
②	Position curve
③	Velocity curve

④	Curve section of a selected command
⑤	Ruler
⑥	Ruler position marking
⑦	Software limit switch position
⑧	Start/stop velocity
⑨	Position axis scale range
⑩	Time axis scale range
⑪	Velocity axis scale range
⑫	Scroll bar, position axis
⑬	Scroll bar time axis
⑭	Scroll bar, velocity axis
⑮	Selecting the grid

Selecting separator sections

If the command table consists of multiple sections separated by separators, you can select these sections in the trend view by selecting a command in the section.

Selecting commands

Commands can be selected in the trend view and in the command table:

- Click on a point on the velocity or position curve in the trend view. The corresponding command will be highlighted in the command table.
- Select a command in the command table.

  The corresponding section of curve will be highlighted.

Selecting the visible range of the trend view

Follow the steps below to adjust the section of the trend view to be displayed:

Select the scaling in the shortcut menu:

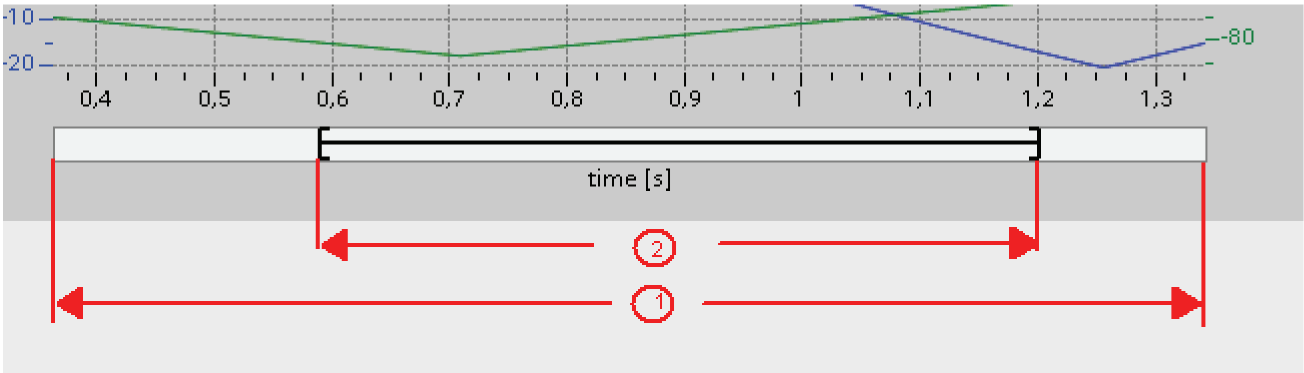
- Scale to curves:

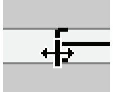
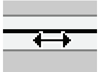
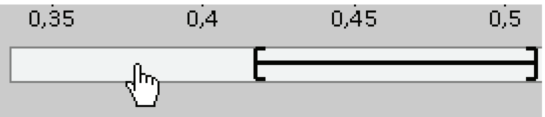
  Scales the axes so the position and velocity curves are visible.
- Scale to curves and limits:

  Scales the axes so the position and velocity curves, the positions of the activated software limit switches and the minimum and maximum velocity limits are visible.

The view selected will be marked in the shortcut menu with a tick.

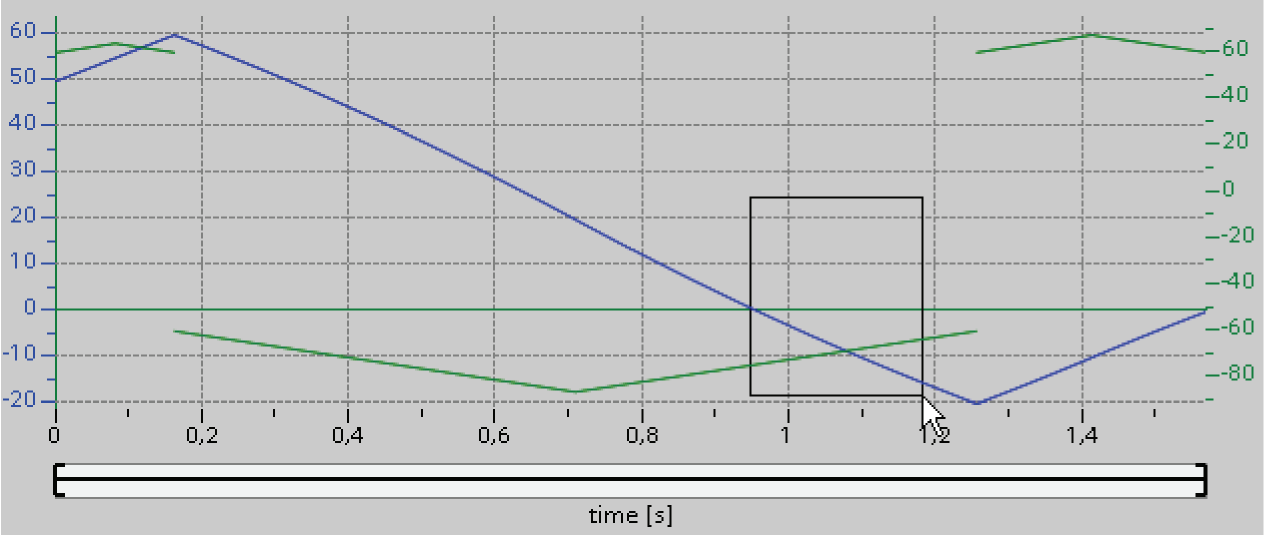
Selecting the section to be shown within the range:



①	Range which the curve values and / or limits are within. (see Selecting in the shortcut menu)
②	Selected range to be shown in the trend window. You set the range with the margin cursor at the right-hand and left-hand margin.  You set the position within range ① with the drag cursor.  You can also define the position by clicking in range ①. 

Selecting the section to be shown with the mouse:

Drag a section of the trend view by clicking and dragging with the mouse. The section of curve selected will be enlarged once you release the mouse.



Undoing the last change to the section:

Select the shortcut command "Undo zoom" to undo the last change to the section.

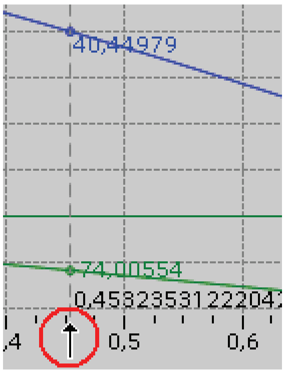
Synchronizing the grid

Click on the axis scales to select whether the grid is to be synchronized with the position axis or velocity axis.

Reading off curve values from the ruler

Activate the ruler using the shortcut menu command "Show ruler".

You can move the ruler to any point on the curves using the ruler cursor.



See also

Configuration - General (Page 2971)

Configuration - Command table (Page 2972)

Shortcut menu commands - Command table (Page 2975)

Shortcut menu commands - Curve chart (Page 2981)

Transition from "Complete command" to "Blend motion" (Page 2982)

Changing the command table configuration in the user program (Page 2983)

Shortcut menu commands - Curve chart

The following shortcut menu commands are available in the curve window:

Zoom 100%

Selects a zoom factor which will show 100% of the curve values and / or limits.

Undo zoom

Undoes the last zoom change.

Scale to curves

Scales the axes so the position and velocity curves are visible.

Scale to curves and limits

Scales the axes so the position and velocity curves, the positions of the activated software limit switches and the minimum and maximum velocity limits are visible.

Show velocity limits

Shows the lines of the velocity limits.

Show software limit switches

Shows the lines of the software limit switches.

Show ruler

Fades the ruler in / out

Use the ruler when you want to see the individual values of the curves.

See also

Configuration - General (Page 2971)

Configuration - Command table (Page 2972)

Shortcut menu commands - Command table (Page 2975)

Working with the trend diagram (Page 2977)

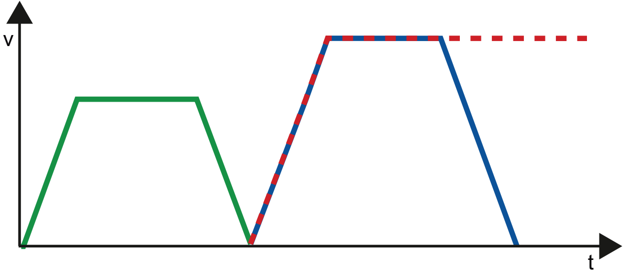
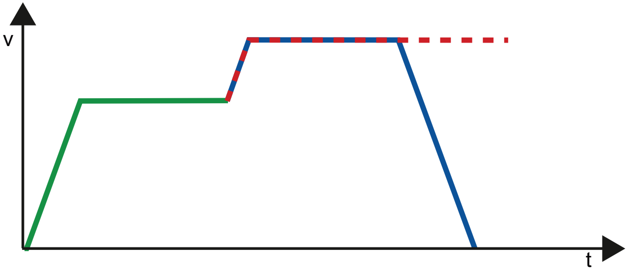
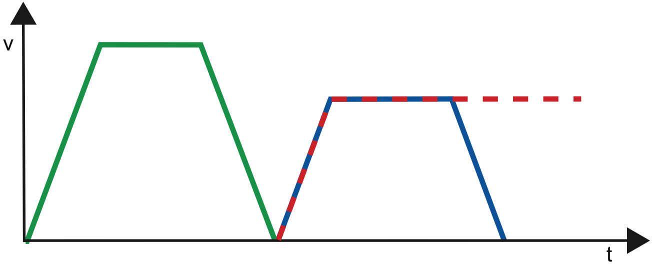
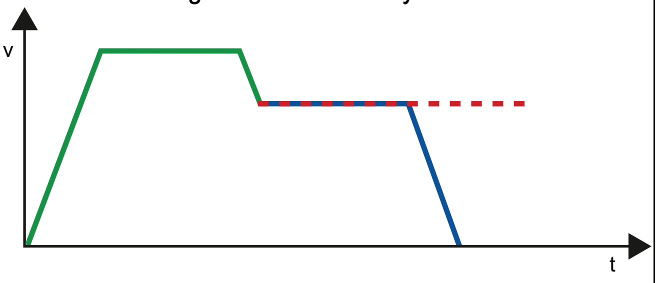
Transition from "Complete command" to "Blend motion" (Page 2982)

Changing the command table configuration in the user program (Page 2983)

Transition from "Complete command" to "Blend motion"

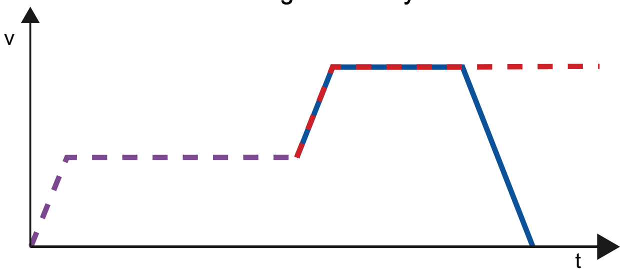
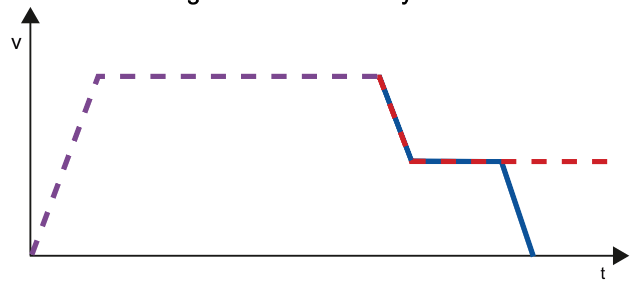
The charts below show the transition between movements in various different transition modes in the "Next step" column:

Motion transition with preceding positioning commands

Complete command	Blend motion
Transition from lower to higher velocity  A job with high velocity is appended to a previous positioning job. The positioning command terminates at its target position at velocity "0". The second command starts from standstill.	Transition from lower to higher velocity  A job with high velocity is overlapped with a previous positioning job. The first positioning command terminates without standstill at its target position. The second command starts with the new velocity.
Transition from higher to lower velocity  A job with low velocity is appended to a previous positioning job. The positioning command terminates at its target position at velocity "0". The second command starts from standstill.	Transition from higher to lower velocity  A job with low velocity is overlapped with a previous positioning job. The first positioning command terminates without standstill at its target position. The first command starts with the new velocity.

🟩	1. Job "Positioning Relative" or "Positioning Absolute"
🟥	2. Job "Velocity set point"
🟦	2. Job "Positioning Relative" or "Positioning Absolute"

Motion transition with preceding velocity commands

Transition from lower to higher velocity:	Transition from higher to lower velocity
	
A command with a high velocity is appended to a previous velocity command. The first velocity command ends after the defined runtime. The second command starts with the new velocity.	A command with low velocity is blended with a previous velocity command. The second command starts with the new velocity.

🟪	1. Command "Velocity set point"
🟥	2. Command "Velocity set point"
🟦	2. Command "Positioning Relative" or "Positioning Absolute"

See also

Configuration - General (Page 2971)

Configuration - Command table (Page 2972)

Shortcut menu commands - Command table (Page 2975)

Working with the trend diagram (Page 2977)

Shortcut menu commands - Curve chart (Page 2981)

Changing the command table configuration in the user program (Page 2983)

Changing the command table configuration in the user program

You can change the following configuration parameters during user program runtime in the CPU:

Jobs and corresponding values

You can also change the parameters of the command table during the runtime of the user program. Use the following technology object variables for this purpose:

- <Table name>.Config.Commands[1..32].Command

  for changing the command type
- <Table name>.Config.Commands[1..32].Position

  for changing the position / travel path
- <Table name>.Config.Commands[1..32].Velocity

  for changing the velocity
- <Table name>.Config.Commands[1..32].Duration

  for changing the duration
- <Table name>.Config.Commands[1..32].BufferMode

  for changing the parameter "Next step"
- <Table name>.Config.Commands[1..32].Code

  for changing the step code

Refer to the description of technology object variables in the Appendix for information on when changes to the configuration parameters take effect.

See also

Configuration - General (Page 2971)

Configuration - Command table (Page 2972)

Shortcut menu commands - Command table (Page 2975)

Working with the trend diagram (Page 2977)

Shortcut menu commands - Curve chart (Page 2981)

Transition from "Complete command" to "Blend motion" (Page 2982)

Extended parameters

Chart parameters

Configuration - General

Configure the basic properties of the chart view of the "Command table" technology object in the "General" configuration window.

Note

If the default axis has been selected under "Use axis parameters of", the unit of measurement can be edited. If a configured axis has been selected, the unit of measurement for this axis will be displayed.

Use axis parameters of

From the drop-down list, select which axis parameters are to be used for selecting the graphic view of and checking the movement sequence. Select "Default axis" if you have yet to add an axis to the "Technology object" folder or wish to use values which have not been configured in any of the available axes.

The axis parameters of the axis selected at the "Axis" parameter will be used to process the command table in the user program.

Unit of measurement

Enter the unit of measurement for the default axis in this field. If a preconfigured axis has been selected under "Use axis parameters of", the unit of measurement configured in these parameter will be displayed.

Configuration - Dynamics

Configure the acceleration and deceleration and the jerk limit for the default axis in the "Dynamics" configuration window.

Note

If the default axis has been selected under "Use axis parameters of", the following fields can be edited. If a configured axis has been selected, the values of this axis will be displayed.

Acceleration / deceleration

Set the desired acceleration of the default axis in the "Acceleration" field. The desired deceleration can be set in the "Deceleration" field.

Motion jobs configured in the command table will be calculated with the selected acceleration / deceleration.

Limit values:

- 1.0e-12 ≤ acceleration ≤ 1.0e12
- 1.0e-12 ≤ deceleration ≤ 1.0e12

Activate jerk limit

Activate the jerk limit with this checkbox.

Step

Set the desired step for ramping up and ramping down in the "Step" field.

Motion jobs configured in the command table will be calculated with the selected step.

Limit values:

- 1.0e-12 ≤ jerk ≤ 1.0e12

Configuration - Limit values

Configure the maximum velocity, the start/stop velocity and the software limit switches of the default axis in the "Limits" configuration window.

Note

If the default axis has been selected under "Use axis parameters of", the following fields can be edited. If a configured axis has been selected, the values of this axis will be displayed.

Maximum velocity / Start/stop velocity

Define the maximum permissible velocity and the start/stop velocity of the default axis in these fields. The start/stop velocity is the minimum permissible velocity of the default axis.

Limit values:

- 1.0e-12 ≤ start/stop velocity ≤ 1.0e12

  Start/stop velocity = 0.0
- 1.0e-12 ≤ maximum velocity ≤ 1.0e12

  Maximum velocity = 0.0

The value of the maximum velocity must be greater or equal to the value of the start/stop velocity.

Enable software limit switches

Activate the function of the low and high software limit switch with this checkbox. Movements in response to software limit switches being reached are not shown in the trend view.

Low / high software limit switch

Enter the position value of the low and high software limit switches in these fields.

Limits:

- -1.0e12 ≤ low software limit switch ≤ -1.0e-12

  1.0e-12 ≤ low software limit switch ≤ 1.0e12

  Low software limit switch = 0.0
- -1.0e12 ≤ high software limit switch ≤ -1.0e-12

  1.0e-12 ≤ high software limit switch ≤ 1.0e12

  High software limit switch = 0.0

The value of the high software limit switch must be greater than or equal to the value of the low software limit switch.

Download to CPU

New or modified project data must be downloaded to the CPU.

Downloading the program blocks and the technology object configuration to the CPU S7-1200

Follow the steps below for downloading:

1. Select the "Program blocks" object or the "Technology objects" object in the project tree.
2. Select the "**Online > Download to device**" menu command.

Note

When blocks are downloaded to the CPU S7-1200, the TIA Portal always ensures afterwards that the offline blocks in the project and the online blocks in the CPU are consistent.

Regardless of the marking in the project tree, all new and modified offline blocks and technology objects are downloaded to the CPU.

On each loading operation, all process values of the data blocks are reset to their initial values.

Downloading of the device configuration to the CPU S7-1200

Follow the steps below for downloading the device configuration:

1. Select the object of the CPU in the project tree
2. Select the "Download to device > Hardware configuration" shortcut menu command.

Downloading the device configuration, the program blocks, and the technology object configuration to the CPU S7-1200

Follow the steps below for downloading:

1. Select the object of the CPU in the project tree
2. Select the "Online > Download to device" menu command.

See also

Guidelines on use of motion control (Page 2938)

Commissioning the axis - Axis control panel

Use the axis command table to move the axis in manual mode, to optimize the axis settings, and to test your system.

The axis control table can only be used if an online connection to the CPU is established.

> NOTICE
>
> **Response times of the axis control panel**
>
> The response time during axis control table operation depends on the communication load of the CPU. Close all other online windows of the TIA Portal to minimize the response time.

"Manual control" button

Click "Manual control" to move the axis in manual control mode. Start by disabling the axis in the user program using motion control instruction "MC_Power". In "Manual control" mode, the axis control table takes over control priority for the axis functions. The user program has no influence on the axis functions until manual control is ended.

> ⚠ WARNING
>
> The Manual control is active for one axis only. A second axis could be moved in Automatic mode, but this would bring about a dangerous situation.
>
> In this case, set the second axis out of operation.

"Automatic mode" button

Click "Automatic mode" to end the "Manual control" mode. The axis control table passes back the control priority and the axis can be controlled by the user program again. The axis must be re-enabled in the user program and homed, if required.

Complete all active traversing motions before switching to automatic control; otherwise, the axis will be braked with the emergency stop deceleration.

"Enable" button

Click "Enable" to enable the axis in "Manual control" mode. When the axis is enabled, the axis control panel functions can be used.

If the axis cannot be enabled because certain conditions are not met, note the error message in the "Error message" field. Information on eliminating errors is available in the Appendix under "List of ErrorIDs and ErrorInfos". After the error has been corrected, enable the axis again.

"Disable" button

Click "Disable" if you want to temporarily disable the axis in "Manual control" mode.

"Command" area

Operation in the "Command" area is only possible if the axis is enabled. You can select one of the following command inputs:

- Jogging

  This command is equivalent to motion control command "MC_MoveJog" in the user program.

- Positioning

  This command is equivalent to the motion control jobs "MC_MoveAbsolute" and "MC_MoveRelative" in the user program. The axis must be homed for absolute positioning.

- Homing

  This command is equivalent to motion control command "MC_Home" in the user program.

  – The "Set reference point" button corresponds to Mode = 0 (direct homing absolute)

  – The "Active homing" button corresponds to Mode = 3 (active homing)

  For active homing, the homing switch must be configured in the axis configuration.

  The values for approach velocity, homing velocity, and reference position offset are taken from the axis configuration unchanged.

Depending on the selection, the relevant fields for entry of setpoints and the buttons for starting the command are displayed.

"Axis status" area

If "Manual control" mode is activated, the current axis status and drive status are shown in the "Axis status" area. The current position and velocity of the axis are displayed at "Process values".

Click "Acknowledge" to acknowledge all cleared errors.

The "Info message" field displays advanced information about the status of the axis.

Error message

The "Error message" field shows the current error. In "Manual control" mode, the error entry can be deleted by pressing the "Acknowledge" button once the error is eliminated.

Note

Initial values for velocity, acceleration / deceleration and jerk

For safety reasons, the "Velocity", "Acceleration/deceleration" and "Jerk" parameters are initialized with values equivalent to only 10% of the configured values when the axis command table is activated. The "Jerk" parameter is only used for technology object "Axis" V2.0 and higher.

The values in the configuration view displayed when you select
"Extended parameters > Dynamics > General" are used for initialization.

The "Velocity" parameter on the control panel is derived from the "Maximum velocity" and the "Acceleration/deceleration" parameters from "Acceleration" in the configuration.

The "Velocity", "Acceleration/deceleration" and "Jerk" parameters can be changed in the axis command table; this does not affect the values in the configuration.

See also

Guidelines on use of motion control (Page 2938)

Working with watch tables (Page 3014)

Programming

Overview of the Motion Control statements

You control the axis with the user program using motion control instructions. The instructions start Motion Control jobs that execute the desired functions.

The status of the motion control jobs and any errors that occur during their execution can be obtained from the output parameters of the Motion Control instructions. The following Motion Control instructions are available:

- MC_Power: Enable, disable axis (Page 1816)
- MC_Reset: Acknowledge error (Page 1821)
- MC_Home: Home axes, set home position (Page 1822)
- MC_Halt: Halt axis (Page 1826)
- MC_MoveAbsolute: Absolute positioning of axes (Page 1829)
- MC_MoveRelative: Relative positioning of axes (Page 1832)
- MC_MoveVelocity: Move axes at preset rotational speed (Page 1836)
- MC_MoveJog: Move axes in jogging mode (Page 1840)
- MC_CommandTable: Run axis jobs as movement sequence (as of technology object "Axis" V2.0) (Page 1844)
- MC_ChangeDynamic: Changing the dynamic settings for the axis (as of technology object "Axis" V2.0) (Page 1846)

See also

Creating a user program (Page 2991)

Programming notes (Page 2994)

Behavior of the Motion Control commands after POWER OFF and restart (Page 2996)

Error displays of the Motion Control statements (Page 3009)

Creating a user program

In the section below you learn how to create a user program with the basic configuration for controlling your axis. All available axis functions are controlled using the Motion Control instructions to be inserted.

Requirement

- The technology object has been created and configured without errors.

Before creating and testing the user program, it is advisable to test the function of the axis and the corresponding parts of the system with the axis command table.

Procedure

Proceed as follows to create the user program in accordance with the principles described below:

1. In the project tree, double-click your code block (the code block must be called in the cyclic program).

   The code block is opened in the programming editor and all available instructions are displayed.
2. Open the "Technology" category and the "Motion Control" and "S7-1200 Motion Control" folders.
3. Use a drag-and-drop operation to move the "MC_Power" instruction to the desired network of the code block.

   The dialog box for defining the instance DB opens.
4. In the next dialog box, select from the following alternatives:

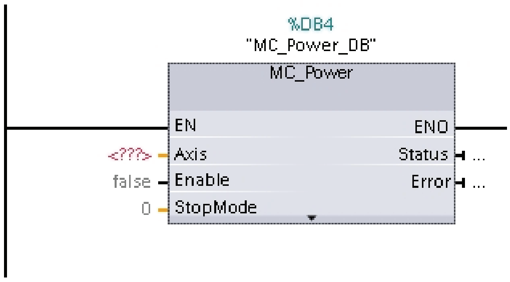
   Single instance

   Click "Single instance" and select whether you want to define the name and number of the instance DB automatically or manually.

   Multi-instance

   Click "Multi-instance" and select whether you want to define the name of the multi-instance automatically or manually.
5. Click "OK".

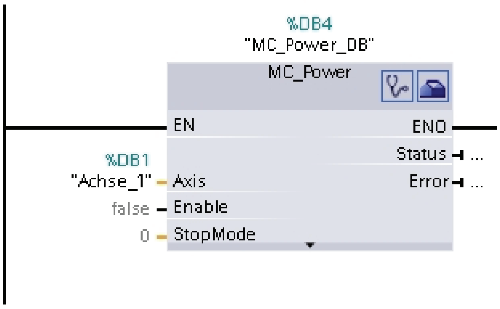
   The Motion Control instruction "MC_Power" is inserted into the network.



Parameters marked with "<???>" must be initialized; all other parameters are assigned default values.

Parameters displayed in black are required for use of the Motion Control instruction.

6. Select technology object in the project tree and drag-and-drop it on <???>.



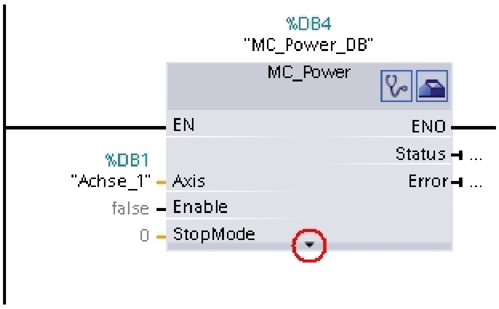
Following selection of the technology object data block, the following buttons are available:

🩺

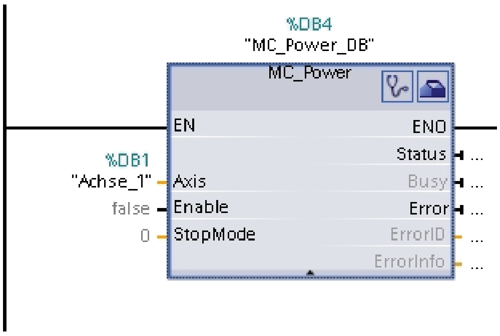
Click the stethoscope icon if you want to open the diagnostics dialog for the technology object.

🧰

Click the toolbox icon if you want to open the configuration view of the technology object.



Click the arrow down icon to view additional parameters of the Motion Control instruction.



The grayed-out parameters now visible can be used optionally.

7. Add your choice of Motion Control instructions in accordance with steps 3 to 6.

Result

You have created the basic configuration for axis control in the user program.

Initialize the input parameters of Motion Control instructions in other parts of the user program to initiate the desired jobs for the "Axis" technology object.

Evaluate the output parameters of the Motion Control instructions and the tags of the data block to track the initiated jobs and the status of the axis.

Refer to the detailed description for details on the parameters of Motion Control instructions.

See also

Overview of the Motion Control statements (Page 2991)

Programming notes (Page 2994)

Behavior of the Motion Control commands after POWER OFF and restart (Page 2996)

Error displays of the Motion Control statements (Page 3009)

Programming notes

When creating your user program, note the following information:

- **Cyclic call of utilized motion control instructions**

  The current status of command execution is available via the output parameters of the motion control instruction. The status is updated with every call of the motion control instruction. Therefore, make sure that the utilized motion control instructions are called cyclically.

- **Transfer of parameter values of a motion control instruction**

  The parameter values pending for the input parameters are transferred with a positive edge at input parameter "Execute" when the block is called.

  The motion control command is started with these parameter values. Parameter values that are subsequently changed for the motion control instruction are not transferred until the next start of the motion control command.

  Exceptions to this are input parameters "StopMode" of motion control instruction "MC_Power" and "Velocity" of motion control instruction "MC_MoveJog". A change in the input parameter is also applied when "Enable" = TRUE, or "JogForward" and "JogBackward". .

- **Programming under consideration of the status information**

  In a stepwise execution of motion control jobs, make sure to wait for the active command to finish before starting a new command. Use the status messages of the motion control instruction and the "StatusBits" tag of the technology object to check for completion of the active command.

  In the examples below, observe the indicated sequence. Failure to observe the sequence will display an axis or command error.

  – **Axis enable with motion control instruction "MC_Power"**

    You must enable the axis before it can take on motion jobs. Use an AND operation of tag <Axis name>.StatusBits.Enable = TRUE with output parameter Status = TRUE of motion control instruction "MC_Power" to verify that the axis is enabled.

  – **Acknowledge error with motion control instruction "MC_Reset"**

    Prior to starting a motion control command, errors requiring acknowledgement must be acknowledged with "MC_Reset". Eliminate the cause of the error and acknowledge the error with motion control instruction "MC_Reset". Verify that the error has been successfully acknowledged before initiating a new command. For this purpose, use an AND operation of tag <Axis name>.StatusBits.Error = FALSE with output parameter Done = TRUE of motion control instruction "MC_Reset".

  – **Home axis with motion control instruction "MC_Home"**

    Before you can start an MC_MoveAbsolute command, the axis must be homed. Use an AND operation of tag <Axis name>.StatusBits.HomingDone = TRUE with output parameter Done = TRUE of motion control instruction "MC_Home" to verify that the axis has been homed.

- **Override of motion control command processing**

  Motion control jobs for moving an axis can also be executed as overriding jobs.

  If a new motion control command is started for an axis while another motion control command is active, the active command is overridden by the new command before the existing command is completely executed. The overridden command signals this using CommandAborted = TRUE in the motion control instruction. It is possible to override an active MC_MoveRelative command with a MC_MoveAbsolute command.

- **Avoiding multiple use of the same instance**

  All relevant information of a motion control command is stored in its instance.

  Do not start a new command using this instance, if you want to track the status of the current command. Use different instances if you want to track the commands separately. If the same instance is used for multiple motion control commands, the status and error information of the individual commands will overwrite each other.

- **Call of motion control instructions in different priority classes (run levels)**

  Motion Control instructions with the same instance may not be called in different priority classes without interlocking. To learn how to call locked motion control instructions, refer to "Tracking commands from higher priority classes (run levels) (Page 3019)".

See also

Overview of the Motion Control statements (Page 2991)

Creating a user program (Page 2991)

Behavior of the Motion Control commands after POWER OFF and restart (Page 2996)

Error displays of the Motion Control statements (Page 3009)

Tracking jobs from higher priority classes (execution levels) (Page 3019)

Behavior of the Motion Control commands after POWER OFF and restart

A POWER OFF or CPU-STOP aborts all active motion control jobs. All CPU outputs, including pulse and direction outputs, are reset.

After a subsequent POWER ON or CPU restart (CPU RUN), the technology objects and the motion control jobs will be reinitialized.

All actual data of the technology objects as well as all status and error information of the previously active motion control jobs are reset to their initial values.

Before the axis can be reused, it must be enabled again using the Motion Control instruction "MC_Power". If homing is required, the axis must be homed again with Motion Control instruction "MC_Home".

See also

Overview of the Motion Control statements (Page 2991)

Creating a user program (Page 2991)

Programming notes (Page 2994)

Error displays of the Motion Control statements (Page 3009)

Monitoring active commands

Monitoring active commands

There are three typical groups for tracking active motion control jobs:

- **Motion control instructions with output parameter "Done"**
- **Motion control instruction "MC_MoveVelocity"**
- **Motion control instruction "MC_MoveJog"**

Motion control instructions with "Done" output parameter

Motion control instructions with the output parameter "Done" are started via input parameter "Execute" and have a defined conclusion (for example, with Motion Control instruction "MC_Home": Homing was successful). The job is complete and the axis is at a standstill.

The jobs of the following Motion Control instructions have a defined conclusion:

- MC_Reset
- MC_Home
- MC_Halt
- MC_MoveAbsolute
- MC_MoveRelative
- MC_CommandTable (as of technology object V2.0)
- MC_ChangeDynamic (as of technology object V2.0)

The output parameter "Done" indicates the value TRUE, if the job has been successfully completed.

The output parameters "Busy", "CommandAborted", and "Error" signal that the job is still being processed, has been aborted or an error is pending. The Motion Control instruction "MC_Reset" cannot be aborted and thus has no "CommandAborted" output parameter. The Motion Control instruction "MC_ChangeDynamic" is completed immediately and therefore has no "Busy" or "CommandAborted" output parameters.

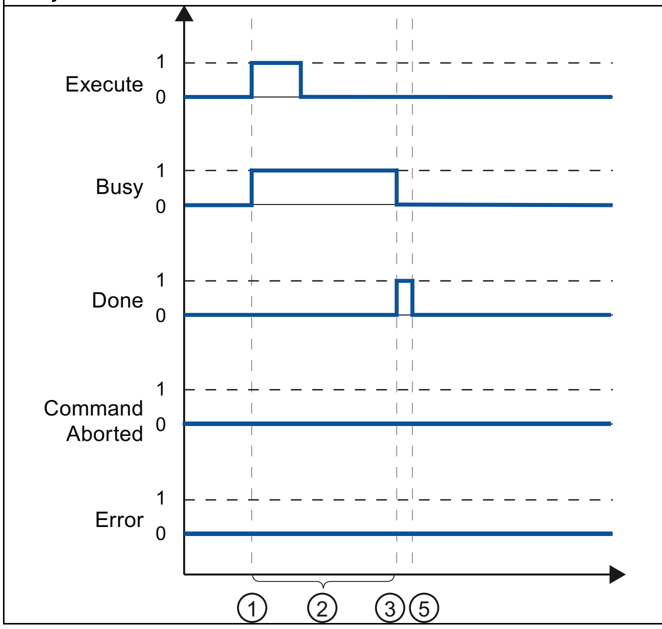
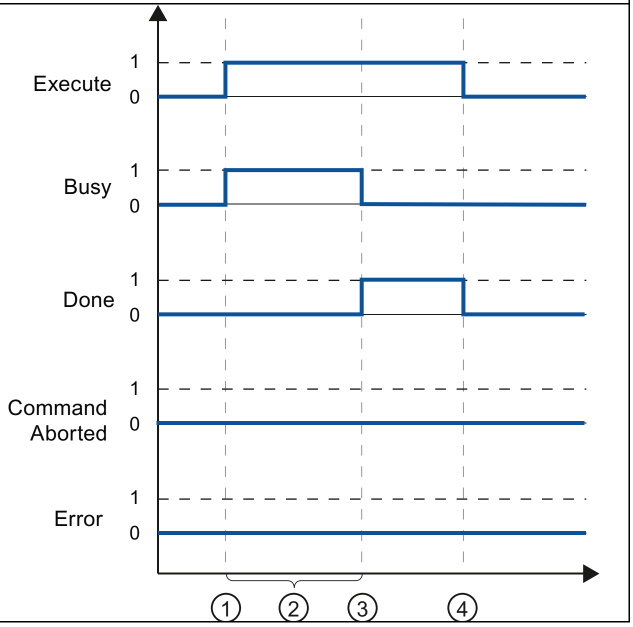
During processing of the motion control job, the output parameter "Busy" indicates the value TRUE. If the job has been completed, aborted, or stopped by an error, the output parameter "Busy" changes its value to FALSE. This change occurs regardless of the signal at input parameter "Execute".

Output parameters "Done", "CommandAborted", and "Error" indicate the value TRUE for at least one cycle. These status messages are latched while input parameter "Execute" is set to TRUE.

The behavior of the status bits is presented below for various example situations:

Complete execution of job

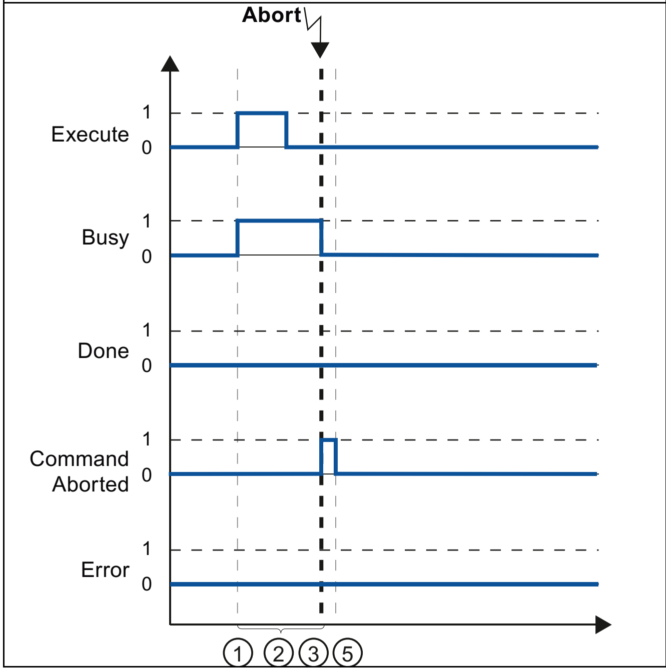
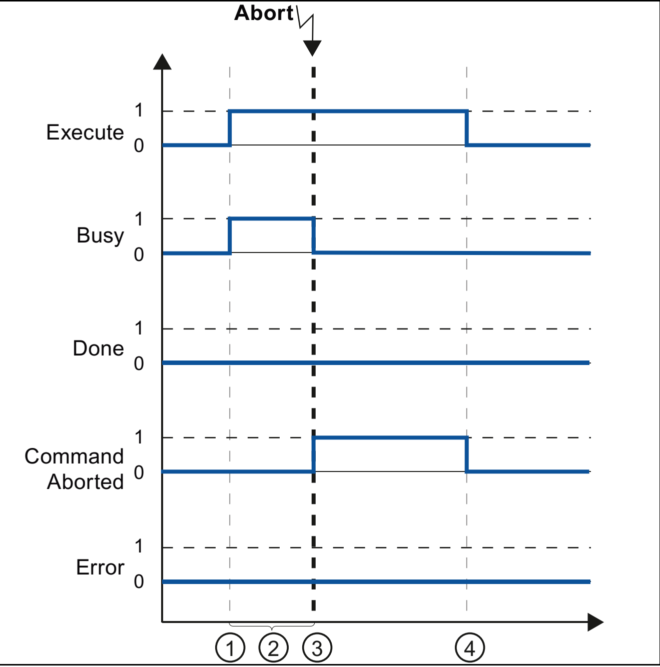
If the motion control job has been completely executed by the time of its conclusion, this is indicated by the value TRUE in output parameter "Done". The signal status of input parameter "Execute" influences the display duration in the output parameter "Done":

"Execute" changes its value to FALSE **during processing** of the job	"Execute" changes its value to FALSE **after completion** of the job
	

①	The job is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the job, or the value TRUE can be retained until after completion of the job.
②	While the job is active, the output parameter "Busy" indicates the value TRUE.
③	With conclusion of the job (for example, for Motion Control instruction "MC_Home": Homing was successful), output parameter "Busy" changes to FALSE and "Done" to TRUE.
④	If "Execute" retains the value TRUE until after completion of the job, then "Done" also remains TRUE and changes its value to FALSE together with "Execute".
⑤	If "Execute" has been set to FALSE before the job is complete, "Done" indicates the value TRUE for only one execution cycle.

Abort job

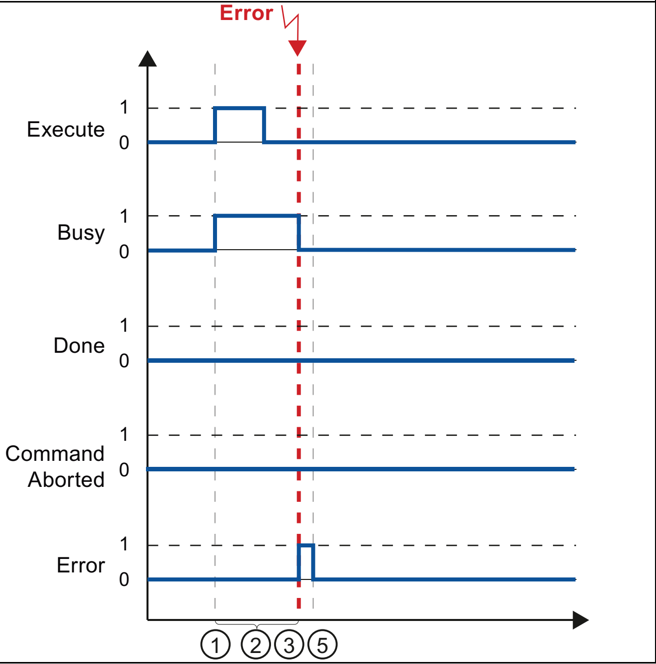
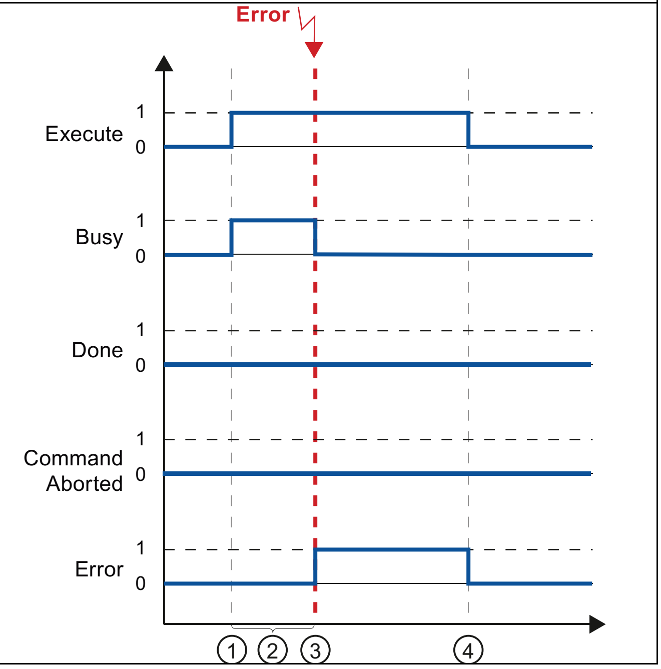
If the motion control job is aborted during execution, this is indicated by the value TRUE in output parameter "CommandAborted". The signal status of the input parameter "Execute" influences the display duration in the output parameter "CommandAborted":

"Execute" changes its value to FALSE **before** the job is aborted.	"Execute" changes its value to FALSE **after** the job is aborted.
 Abort Execute 1 0 Busy 1 0 Done 1 0 Command Aborted 1 0 Error 1 0 ① ② ③ ⑤	 Abort Execute 1 0 Busy 1 0 Done 1 0 Command Aborted 1 0 Error 1 0 ① ② ③ ④

①	The job is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the job, or the value TRUE can be retained until after completion of the job.
②	While the job is active, the output parameter "Busy" indicates the value TRUE.
③	During job execution, the job is aborted by another motion control job. If the job is aborted, output parameter "Busy" changes to FALSE and "CommandAborted" to TRUE.
④	If "Execute" retains the value TRUE until after the job is aborted, then "CommandAborted" also remains TRUE and changes its value to FALSE together with "Execute".
⑤	If "Execute" has been set to FALSE before the job is aborted, "CommandAborted" indicates the value TRUE for only one execution cycle.

Error during job execution

If an error occurs during execution of the motion control job, this is indicated by the value TRUE in the output parameter "Error". The signal status of the input parameter "Execute" influences the display duration in the output parameter "Error":

"Execute" changes its value to FALSE **before** the error occurs	"Execute" changes its value to FALSE **after** the error occurs
	

①	The job is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the job, or the value TRUE can be retained until after completion of the job.
②	While the job is active, the output parameter "Busy" indicates the value TRUE.
③	An error occurred during job execution. When the error occurs, the output parameter "Busy" changes to FALSE and "Error" to TRUE.
④	If "Execute" retains the value TRUE until after the error occurs, then "Error" also remains TRUE and only changes its value to FALSE together with "Execute".
⑤	If "Execute" has been set to FALSE before the error occurs, "Error" indicates the value TRUE for only one execution cycle.

Motion control instruction MC_MoveVelocity

The jobs of Motion Control instruction "MC_MoveVelocity" do not have a defined end. The job objective is fulfilled when the parameterized velocity is reached for the first time and the axis travels at constant velocity. When the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "InVelocity".

The job is complete when the parameterized velocity has been reached and input parameter "Execute" has been set to the value FALSE. However, the axis motion is not yet complete upon completion of the job. For example, the axis motion can be stopped with motion control job "MC_Halt".

The output parameters "Busy", "CommandAborted", and "Error" signal that the job is still being processed, has been aborted or an error is pending.

During execution of the motion control job, output parameter "Busy" indicates the value TRUE. If the job has been completed, aborted, or stopped by an error, the output parameter "Busy" changes its value to FALSE. This change occurs regardless of the signal at input parameter "Execute".

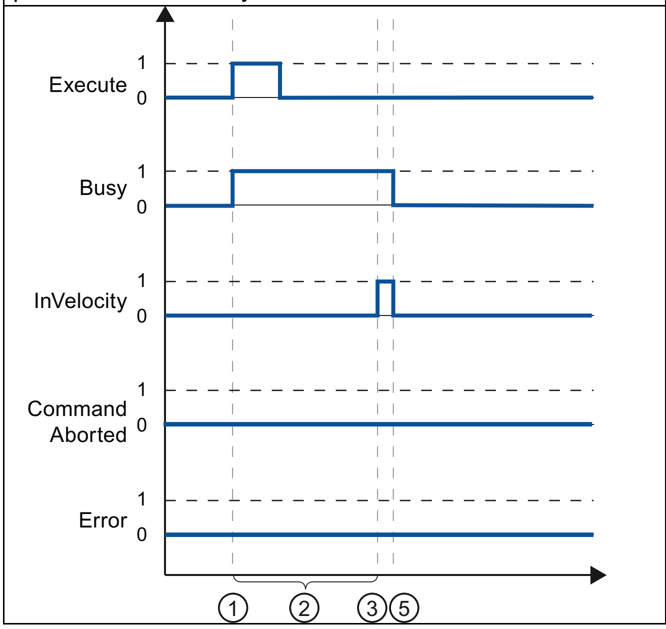
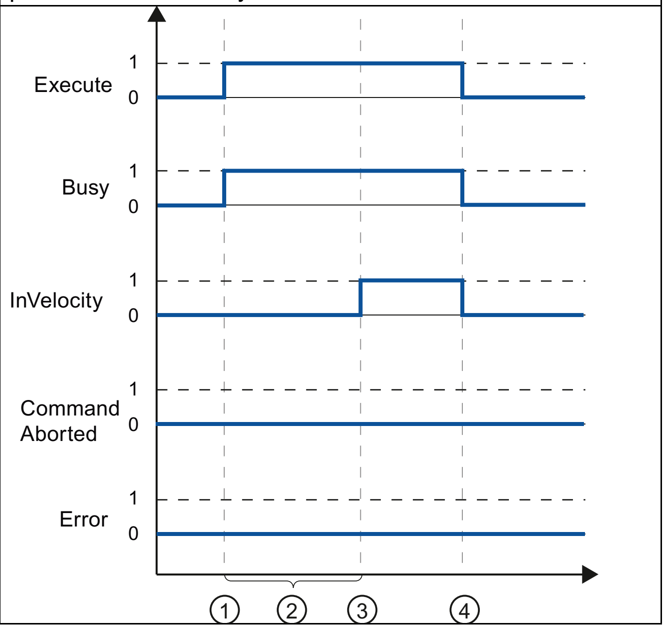
The output parameters "InVelocity", "CommandAborted", and "Error" indicate the value TRUE for at least one cycle, when their conditions are met. These status messages are latched while input parameter "Execute" is set to TRUE.

The behavior of the status bits is presented below for various example situations:

The parameterized velocity is reached

If the motion control job has been executed by the time the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "InVelocity".

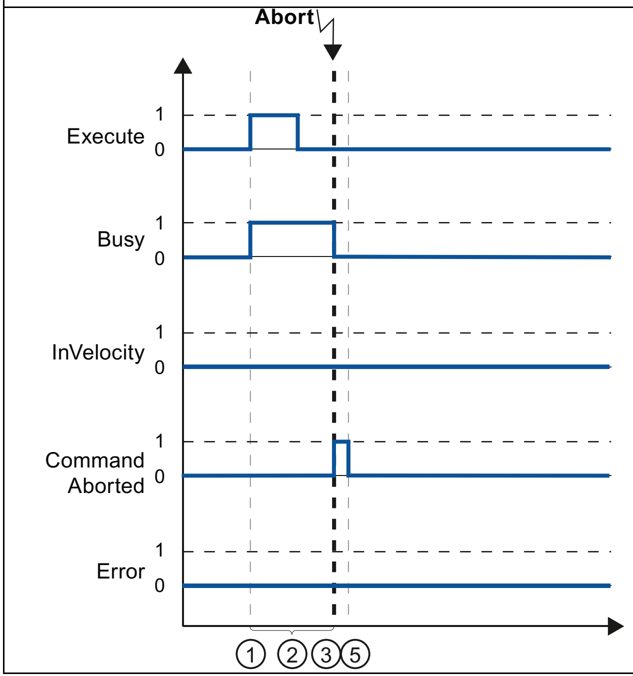
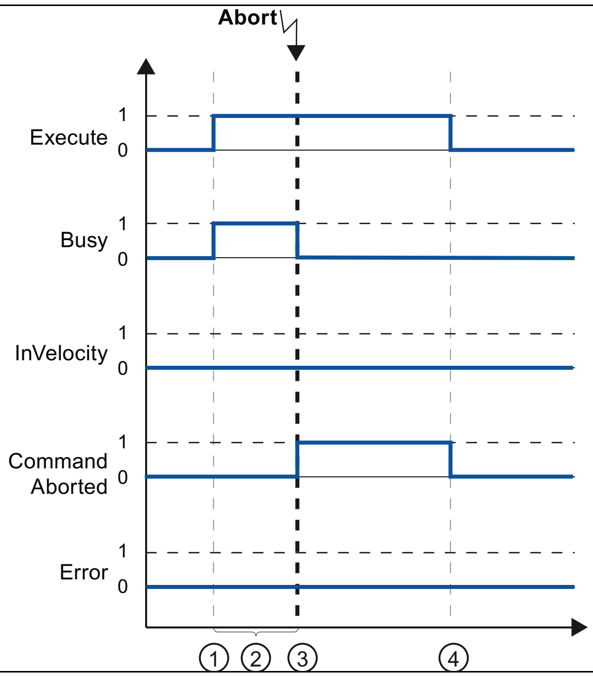
The signal status of the input parameter "Execute" influences the display duration in the output parameter "InVelocity":

"Execute" changes its value to FALSE **before** the parameterized velocity is reached	"Execute" changes its value to FALSE **after** the parameterized velocity is reached
	

①	The job is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can be reset to the value FALSE event before the parameterized velocity is reached, or alternatively only after it has been reached.
②	While the job is active, the output parameter "Busy" indicates the value TRUE.
③	When the parameterized velocity is reached, the output parameter "InVelocity" changes to TRUE.
④	If "Execute" retains the value TRUE even after the parameterized velocity has been reached, the job remains active. "InVelocity" and "Busy" retain the value TRUE and only change their status to FALSE together with "Execute".
⑤	If "Execute" has been reset to FALSE before the parameterized velocity is reached, the job is complete when the parameterized velocity is reached. "InVelocity" indicates the value TRUE for one execution cycle and changes to FALSE together with "Busy".

The job is aborted prior to reaching the parameterized velocity

If the motion control job is aborted before the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "CommandAborted". The signal status of input parameter "Execute" influences the display duration in output parameter "CommandAborted".

"Execute" changes its value to FALSE **before** the job is aborted.	"Execute" changes its value to FALSE **after** the job is aborted.
 Abort Execute 1 0 Busy 1 0 InVelocity 1 0 Command Aborted 1 0 Error 1 0 ① ② ③ ⑤	 Abort Execute 1 0 Busy 1 0 InVelocity 1 0 Command Aborted 1 0 Error 1 0 ① ② ③ ④

①	The job is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the job, or the value TRUE can be retained until after the job is aborted.
②	While the job is active, the output parameter "Busy" indicates the value TRUE.
③	During job execution, the job is aborted by another motion control job. If the job is aborted, output parameter "Busy" changes to FALSE and "CommandAborted" to TRUE.
④	If "Execute" retains the value TRUE until after the job is aborted, then "CommandAborted" also remains TRUE and changes its status to FALSE together with "Execute".
⑤	If "Execute" has been reset to FALSE before the job is aborted, "CommandAborted" indicates the value TRUE for only one execution cycle.

Note

Under the following conditions, an abort is not indicated in output parameter "CommandAborted":

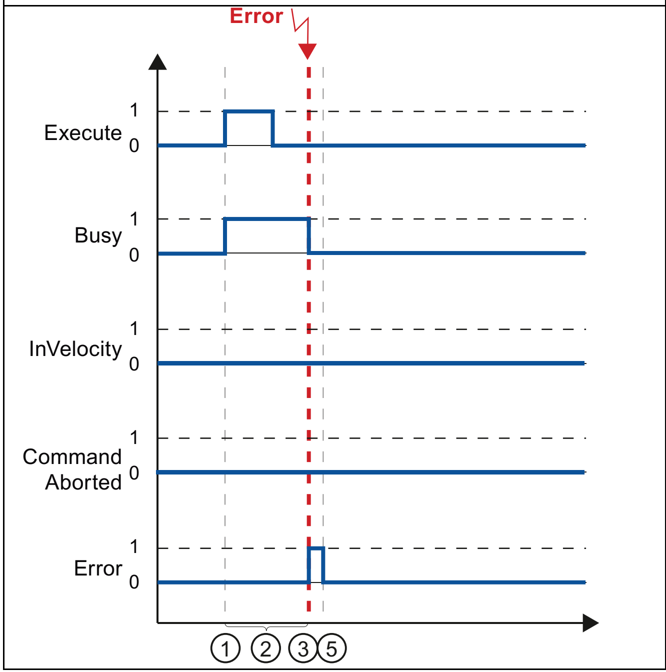
The parameterized velocity has been reached, input parameter "Execute" has the value FALSE, and a new motion control job is initiated.

When the parameterized velocity is reached and input parameter "Execute" has the value FALSE, the job is complete. Therefore, the start of a new job is not indicated as an abort.

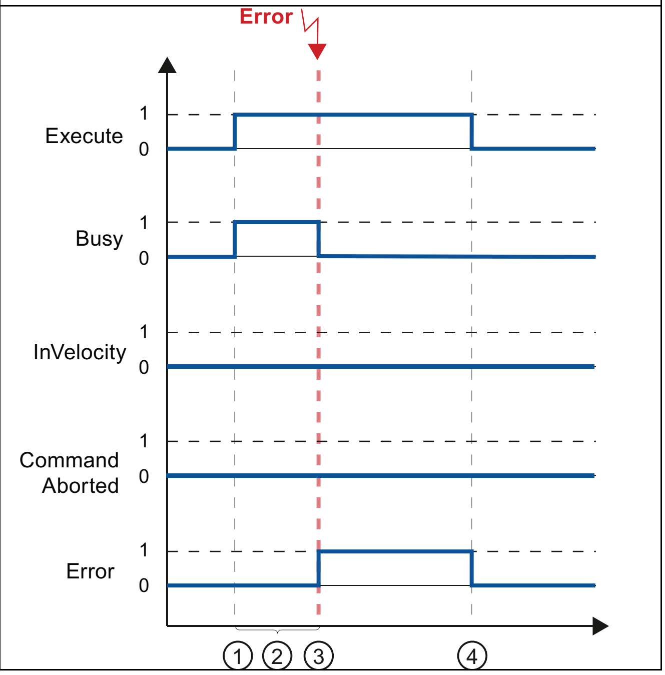
An error has occurred prior to reaching the parameterized velocity

If an error occurs during execution of the motion control job before the parameterized velocity has been reached, this is indicated by the value TRUE in the output parameter "Error". The signal status of the input parameter "Execute" influences the display duration in the output parameter "Error":

"Execute" changes its value to FALSE **before** the error occurs



"Execute" changes its value to FALSE **after** the error occurs



①	The job is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the job, or the value TRUE can be retained until after the error has occurred.
②	While the job is active, the output parameter "Busy" indicates the value TRUE.
③	An error occurred during job execution. When the error occurs, the output parameter "Busy" changes to FALSE and "Error" to TRUE.
④	If "Execute" retains the value TRUE until after the error has occurred, then "Error" also remains TRUE and only changes its status to FALSE together with "Execute".
⑤	If "Execute" has been reset to FALSE before the error occurs, "Error" indicates the value TRUE for only one execution cycle.

Note

Under the following conditions, an error is not indicated in output parameter "Error":

The parameterized velocity has been reached, input parameter "Execute" has the value FALSE, and an axis error occurs (software limit switch is approached, for example).

When the parameterized velocity is reached and input parameter "Execute" has the value FALSE, the job is complete. After completion of the job, the axis error is only indicated in the Motion Control instruction "MC_Power".

Motion control instruction MC_MoveJog

The jobs of Motion Control instruction "MC_MoveJog" implement a jog operation.

The motion control jobs "MC_MoveJog" do not have a defined end. The job objective is fulfilled when the parameterized velocity is reached for the first time and the axis travels at constant velocity. When the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "InVelocity".

The order is complete when input parameter "JogForward" or "JogBackward" has been set to the value FALSE and the axis has come to a standstill.

The output parameters "Busy", "CommandAborted", and "Error" signal that the job is still being processed, has been aborted or an error is pending.

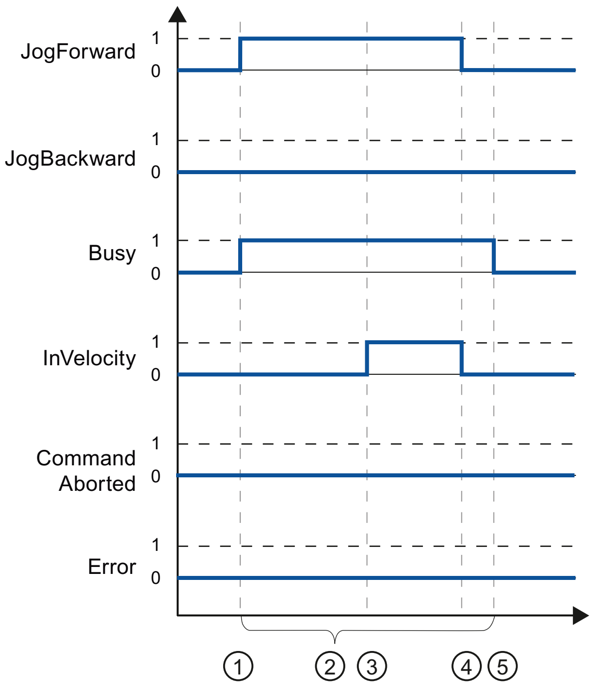
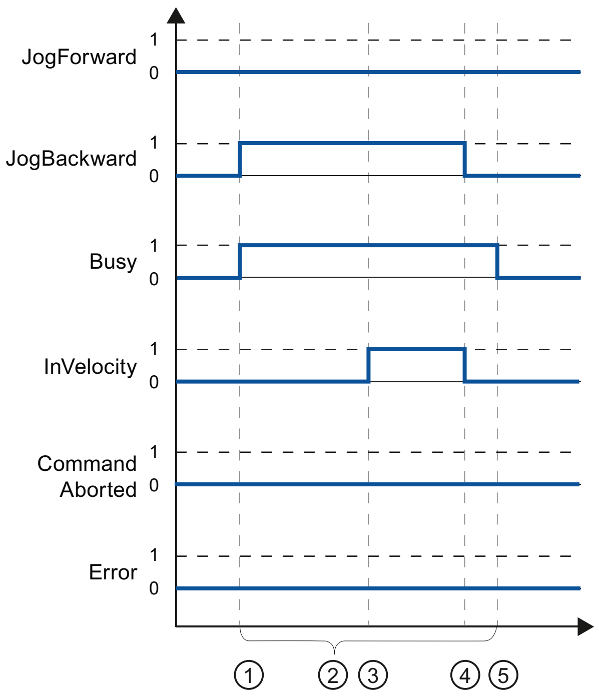
During processing of the motion control job, the output parameter "Busy" indicates the value TRUE. If the job has been completed, aborted, or stopped by an error, the output parameter "Busy" changes its value to FALSE.

The output parameter "InVelocity" indicates the status TRUE, as long as the axis is moving at the parameterized velocity. The output parameters "CommandAborted" and "Error" indicate the status for at least one cycle. These status messages are latched as long as either input parameter "JogForward" or "JogBackward" is set to TRUE.

The behavior of the status bits is presented below for various example situations:

The parameterized velocity is reached and maintained

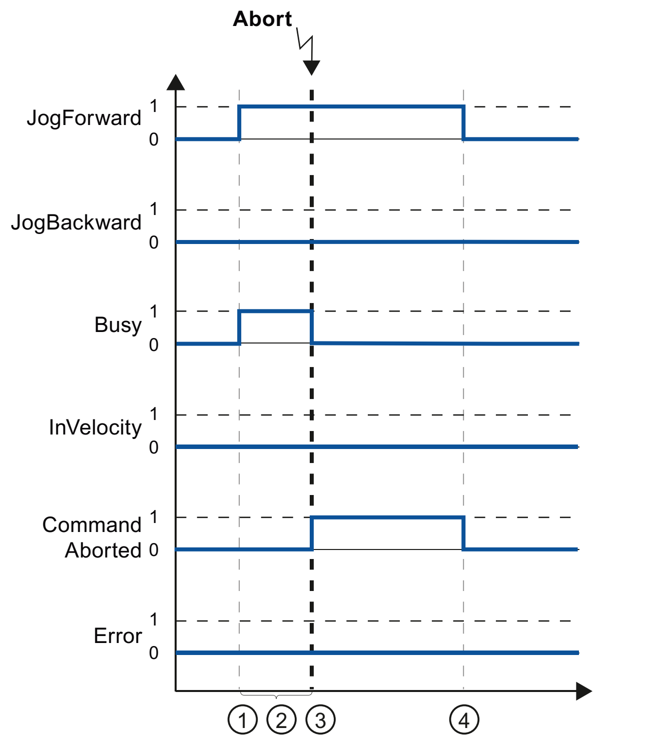
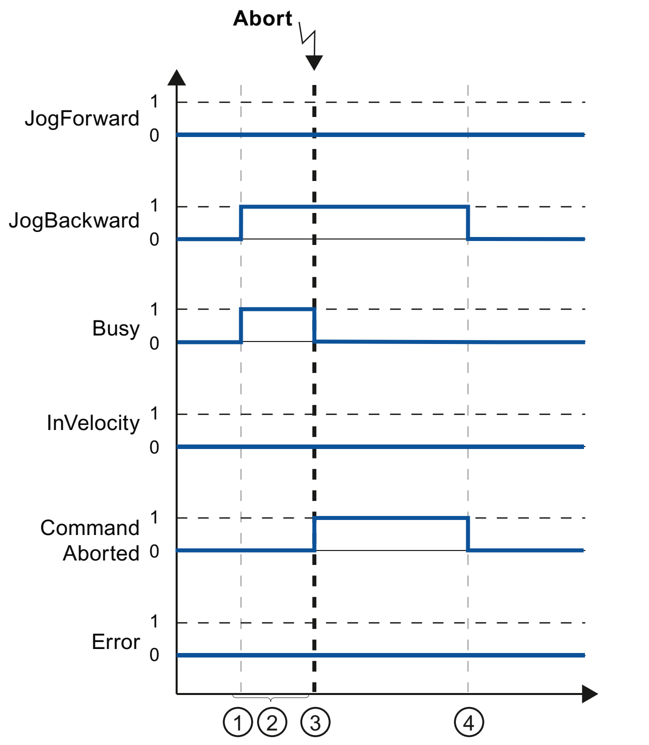
If the motion control job has been executed by the time the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "InVelocity".

Jog mode is controlled by input parameter "JogForward"	Jog mode is controlled by input parameter "JogBackward".
 JogForward JogBackward Busy InVelocity Command Aborted Error 1 0 ① ② ③ ④ ⑤	 JogForward JogBackward Busy InVelocity Command Aborted Error 1 0 ① ② ③ ④ ⑤

①	The job is started with a positive edge at the input parameter "JogForward" or "JogBackward".
②	While the job is active, the output parameter "Busy" indicates the value TRUE.
③	When the parameterized velocity is reached, the output parameter "InVelocity" changes to TRUE.
④	When the input parameter "JogForward" or "JogBackward" is reset to the value FALSE, the axis motion ends. The axis starts to decelerate. As a result, the axis no longer moves at constant velocity and the output parameter "InVelocity" changes its status to FALSE.
⑤	If the axis has come to a standstill, the motion control job is complete and the output parameter "Busy" changes its value to FALSE.

The job is aborted during execution

If the motion control job is aborted during execution, this is indicated by the value TRUE in output parameter "CommandAborted". The behavior is independent of whether or not the parameterized velocity has been reached.

Jog mode is controlled by input parameter "JogForward".	Jog mode is controlled by input parameter "JogBackward".
	

①	The job is started with a positive edge at the input parameter "JogForward" or "JogBackward".
②	While the job is active, the output parameter "Busy" indicates the value TRUE.
③	During job execution, the job is aborted by another motion control job. If the job is aborted, output parameter "Busy" changes to FALSE and "CommandAborted" to TRUE.
④	When the input parameter "JogForward" or "JogBackward" is reset to the value FALSE, the output parameter "CommandAborted"" changes its value to FALSE.

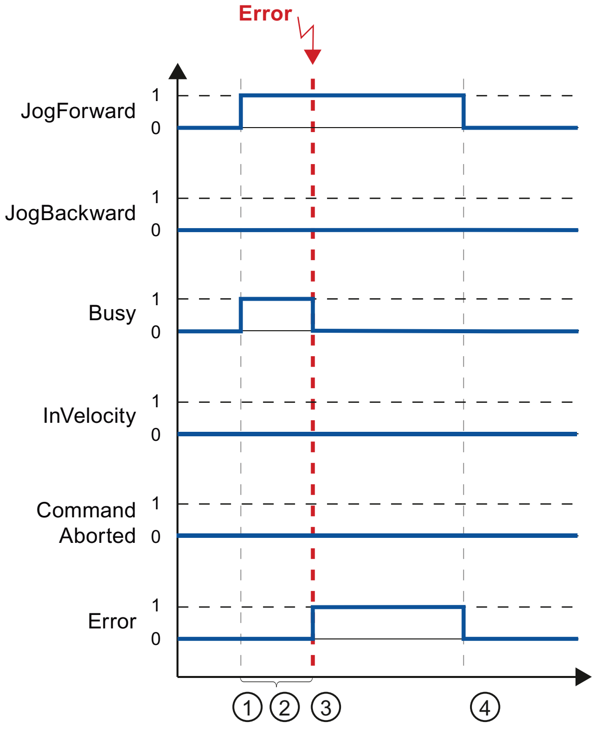
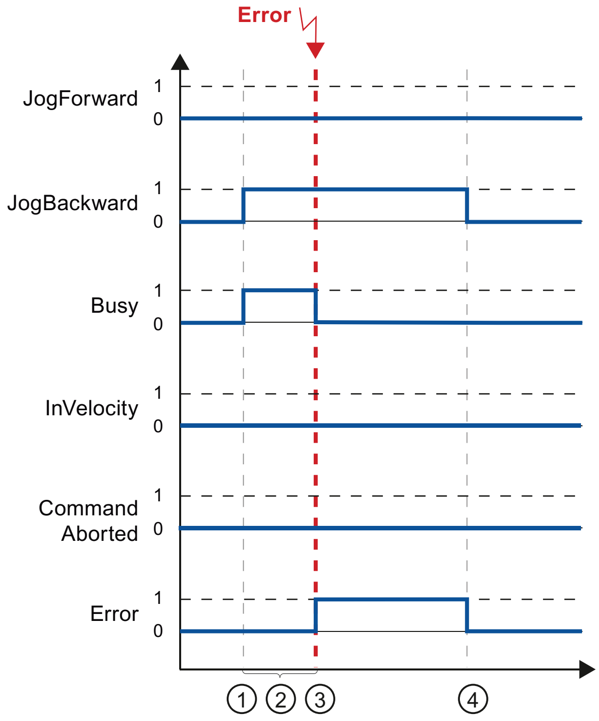
Note

The job abort is indicated in the output parameter "CommandAborted" for only one execution cycle, if all conditions below are met:

The input parameters "JogForward" and "JogBackward" have the value FALSE (but the axis is still decelerating) and a new motion control job is initiated.

An error has occurred during job execution

If an error occurs during execution of the motion control job, this is indicated by the value TRUE in output parameter "Error". The behavior is independent of whether or not the parameterized velocity has been reached.

Jog mode is controlled by input parameter "JogForward".	Jog mode is controlled by input parameter "JogBackward".
	

①	The job is started with a positive edge at the input parameter "JogForward" or "JogBackward".
②	While the job is active, the output parameter "Busy" indicates the value TRUE.
③	An error occurred during job execution. When the error occurs, the output parameter "Busy" changes to FALSE and "Error" to TRUE.
④	When the input parameter "JogForward" or "JogBackward" is reset to the value FALSE, the output parameter "Error" changes its value to FALSE.

Note

An error occurrence is indicated in the output parameter "Error" for only one execution cycle, if all the conditions below are met:

The input parameters "JogForward" and "JogBackward" have the value FALSE (but the axis is still decelerating) and a new error occurs (software limit switch is approached, for example).

Error displays of the Motion Control statements

The Motion Control instructions indicate any motion control job and technology object errors in the output parameters "Error", "ErrorID" and "ErroInfo"of the Motion Control instructions.

Error display at output parameters "Error", "ErrorID" and "ErrorInfo"

If the output parameter "Error" indicates the value TRUE, the complete job, or portions thereof, could not be executed. The cause of the error is indicated by the value in output parameter "ErrorID" . Detailed information about the cause of the error is returned by the value in output parameter ErrorInfo". We distinguish between the following error classes for error indication:

- Operating error with axis stop (for example, "HW limit switch was approached")

  Operating errors with axis stop are errors that occur during runtime of the user program. If the axis is in motion, it is stopped with the configured deceleration or emergency stop deceleration, depending on the error. The errors are indicated in the error-triggering Motion Control instruction and in the Motion Control instruction "MC_Power".

- Operating error without axis stop (for example, "Axis is not homed")

  Operating errors without axis stop are errors that occur during runtime of the user program. If the axis is in motion, the motion is continued. The errors are only indicated in the Motion Control instruction which triggers the error.

- Parameterization error of Motion Control instruction "
  (for example, "Incorrect value in parameter "Velocity"")

  Parameterization errors occur when incorrect information is specified in the input parameters of Motion Control instructions. If the axis is in motion, the motion is continued. The errors are only indicated in the Motion Control instruction which triggers the error.

- Configuration error on technology object "Axis" (for example, "Value for "Acceleration" is invalid")

  A configuration error exists if one or more parameters are incorrectly configured in the axis configuration or if editable configuration data have been modified incorrectly during runtime of the program. An axis in motion is stopped with the configured emergency stop deceleration. The error is indicated in the error-triggering Motion Control instruction and in Motion Control instruction "MC_Power".

- **Configuration error on technology object "Command table" (for example "Value for "Velocity" is invalid")**

  There is a configuration error if one or more parameters are incorrectly set in the axis command table or if programmable configuration data have been modified incorrectly during runtime of the program. If the axis is in motion, the motion is continued. The errors are only indicated in the "MC_CommandTable" Motion Control instruction.

- **Internal error**

  When an internal error occurs, the axis is stopped. The errors are indicated in the error-triggering Motion Control instruction and, in some cases, in the Motion Control instruction "MC_Power".

A detailed description of the ErrorIDs and ErrorInfos, as well as their remedies, is available in the Appendix.

See also

Overview of the Motion Control statements (Page 2991)

Creating a user program (Page 2991)

Programming notes (Page 2994)

Behavior of the Motion Control commands after POWER OFF and restart (Page 2996)

Axis - Diagnostics

Status and error bits

You use the "Status and error bits" diagnostic function to monitor the most important status and error messages for the axis in the TIA Portal. The diagnostic function display is available in online mode in "Manual control" mode and in "Automatic control" when the axis is active. The status error messages have the following meaning:

Status of the axis

Status	Description
Enabled	The axis is enabled and ready to be controlled via motion control jobs. (Tag of technology object: <Axis name>.StatusBits.Enable)
Homed	The axis is homed and is capable of executing absolute positioning jobs of Motion Control instruction "MC_MoveAbsolute". The axis does not have to be homed for relative homing. Special situations: • During active homing, the status is FALSE. • If a homed axis undergoes passive homing, the status is set to TRUE during passive homing. (Tag of technology object: <Axis name>.StatusBits.HomingDone)

Status	Description
Axis error	An error has occurred in the "Axis" technology object. More information about the error is available in automatic control at the ErrorID and ErrorInfo parameters of the Motion Control instructions. In manual mode, the "Error message" field of the axis command table displays detailed information about the cause of error. (Tag of technology object: <Axis name>.StatusBits.Error)
Axis command table enabled	The "Manual control" mode was enabled in the axis command table. The axis command table has control priority over the "Axis" technology object. The axis cannot be controlled from the user program. (Tag of technology object: <Axis name>.StatusBits.ControlPanelActive)

Drive status

Status	Description
Drive ready	The drive is ready for operation. (Tag of technology object: <Axis name>.StatusBits.DriveReady)
Drive error	The drive has reported an error after failure of its "Drive ready" signal. (Tag of technology object: <Axis name>.ErrorBits.DriveFault)

Status of the axis motion

Status	Description
Standstill	The axis is at a standstill. (Tag of technology object: <Axis name>.StatusBits.StandStill)
Accelerating	The axis accelerates. (Tag of technology object: <Axis name>.StatusBits.Acceleration)
Constant velocity	The axis travels at constant velocity. (Tag of technology object: <Axis name>.StatusBits.ConstantVelocity)
Decelerating	The axis decelerates (slows down). (Tag of technology object: <Axis name>.StatusBits.Deceleration)

Status of the motion mode

Status	Description
Positioning	The axis executes a positioning job of the Motion Control instruction "MC_MoveAbsolute" or "MC_MoveRelative" or of the axis command table. (Tag of technology object: <Axis name>.StatusBits.PositioningCommand)
Travel with velocity specification	The axis executes a job with the velocity specification of the Motion Control instruction "MC_MoveVelocity" or "MC_MoveJog" or of the axis command table. (Tag of technology object: <Axis name>.StatusBits.SpeedCommand)

Status	Description
Homing	The axis executes a homing job of the Motion Control instruction "MC_Home" or the axis command table. (Tag of technology object: <Axis name>.StatusBits.Homing)
Command table active (as of technology object Axis V2.0)	The axis is controlled by Motion Control instruction "MC_CommandTable". (Technology object tag: <Axis name>.StatusBits.CommandTableActive)

Error messages

Error	Description
Low software limit switch has been reached	The low software limit switch has been reached. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMinReached)
Low software limit switch has been exceeded	The low software limit switch has been exceeded. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMinExceeded)
High software limit switch has been reached	The high software limit switch has been reached. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMaxReached)
High software limit switch has been exceeded	The high software limit switch has been exceeded. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMaxExceeded)
Low hardware limit switch was approached	The low hardware limit switch has been approached. (Tag of technology object: <Axis name>.ErrorBits.HwLimitMin)
High hardware limit switch has been approached	The high hardware limit switch has been approached. (Tag of technology object: <Axis name>.ErrorBits.HwLimitMax)
PTO and HSC already in use	A second axis is using the same PTO (Pulse Train Output) and HSC (High Speed Counter) and is enabled with "MC_Power". (Tag of technology object: <Axis name>.ErrorBits.HwUsed)
Configuration error	The "Axis" technology object was incorrectly configured or editable configuration data were modified incorrectly during runtime of the user program. (Tag of technology object: <Axis name>.ErrorBits.ConfigFault)
Internal error	An internal error has occurred. (Tag of technology object: <Axis name>.ErrorBits.SystemFault)

See also

Motion status (Page 3013)

StatusBits. tag (Page 3048)

ErrorBits. tag (Page 3051)

Motion status

Use the "Motion status" diagnostic function to monitor the motion status of the axis in the TIA Portal. The diagnostic function display is available in online mode in "Manual control" mode and in "Automatic control" when the axis is active. The displayed status information has the following meaning:

Status	Description
Position	The "Position" field indicates the current axis position. If the axis is not homed, the value indicates the position value relative to the enable position of the axis. (Tag of technology object: <Axis name>.MotionStatus.Position)
Velocity	The "Velocity" field indicates the current axis velocity. (Tag of technology object: <Axis name>.MotionStatus.Velocity)
Target position	The "Target position" field indicates the current target position of an active positioning job or of the axis command table. The value of the "Target position" is only valid during execution of a positioning job. (Tag of technology object: <Axis name>.MotionStatus.TargetPosition)
Remaining travasing distance	The "Remaining traversing distance" field indicates the traversing distance currently remaining for an active positioning job or the axis command table. The "Remaining travasing distance" value is only valid during execution of a positioning job. (Tag of technology object: <Axis name>.MotionStatus.Distance)

See also

Status and error bits (Page 3010)

MotionStatus. tag (Page 3047)

Dynamics settings

Use the "Dynamics settings" diagnostic function to monitor the dynamic limit values of the axis in the TIA Portal. The diagnostic function display is available in online mode in "Manual control" mode and in "Automatic control" when the axis is active. The status information displayed has the following meaning:

Dynamic limit	Description
Acceleration	The "Acceleration" field indicates the currently configured acceleration of the axis. (Technology object variable: <Axis name>.Config.DynamicDefaults.Acceleration)
Deceleration	The "Deceleration" field indicates the currently configured deceleration of the axis. (Technology object variable: <Axis name>.Config.DynamicDefaults.Deceleration)
Emergency stop deceleration	The "Emergency stop deceleration" field indicates the currently configured emergency stop deceleration of the axis. (Technology object variable: <Axis name>.Config.DynamicDefaults.EmergencyDeceleration)
Step (as of technology object Axis V2.0)	The "Velocity" field indicates the current axis step velocity configured. (Technology object variable: <Axis name>.Config.DynamicDefaults.Jerk)

Working with watch tables

Use watch tables if you want to monitor and modify tags of motion control instructions or the "Axis" technology object during commissioning.

To monitor and modify tags, you must specify the complete name of the tag, including object name and all structure names in a watch table.

Example: <Axis name>.Config.DynamicDefaults.Acceleration)

Tip:

You can use a copy & paste operation to avoid entering long tag names.

Procedure

To insert the tag names, follow the steps described below:

1. In the project tree, select the instance data block or the technology object of the axis.
2. **Parameters of the motion control instruction**
   - Right-click and select the **Open** command in the shortcut menu.

   Tags of the technology object
   - Right-click and select the **Open in editor** command in the shortcut menu.
3. **Parameters of the motion control instruction**
   - Select the lines of the tags in the Input or Output area

   Tags of the technology object
   - In the Static area, open the relevant structures and select the lines of the tags
4. Select the **Edit > Copy** menu command.
5. Double-click to open the watch table.
6. Select the line starting at which the tags are to be inserted
7. Select the **Edit > Paste** menu command.

Insert the tags with their complete names in the watch table.

⚠ WARNING

The watch table also gives you write access to tags whose use is blocked for safety reasons in the user program. Modifying these tags can result in damage to the current axis configuration and to undefined responses of the axis. Only modify those tags whose access is marked with "RW" in the tag list of the technology object.

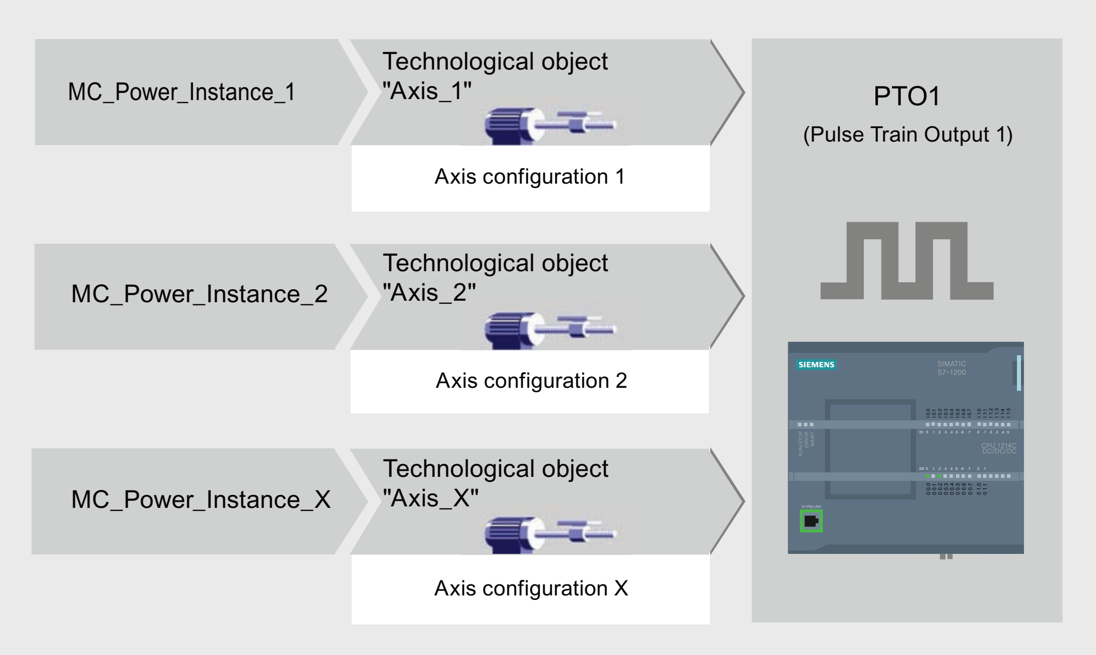
See also

Commissioning the axis - Axis control panel (Page 2988)

Appendix

Using multiple axes with the same PTO

Use the motion control functionality of the CPU S7-1200 to run multiple "Axis" technology objects with the same PTO (Pulse Train Output) and thus with the same CPU outputs. This is appropriate, for example, if different axis configurations are to be used for different production sequences via one PTO. As described below, it is possible to switch between these axis configurations as often as necessary. The following diagram presents the basic functional relationships:

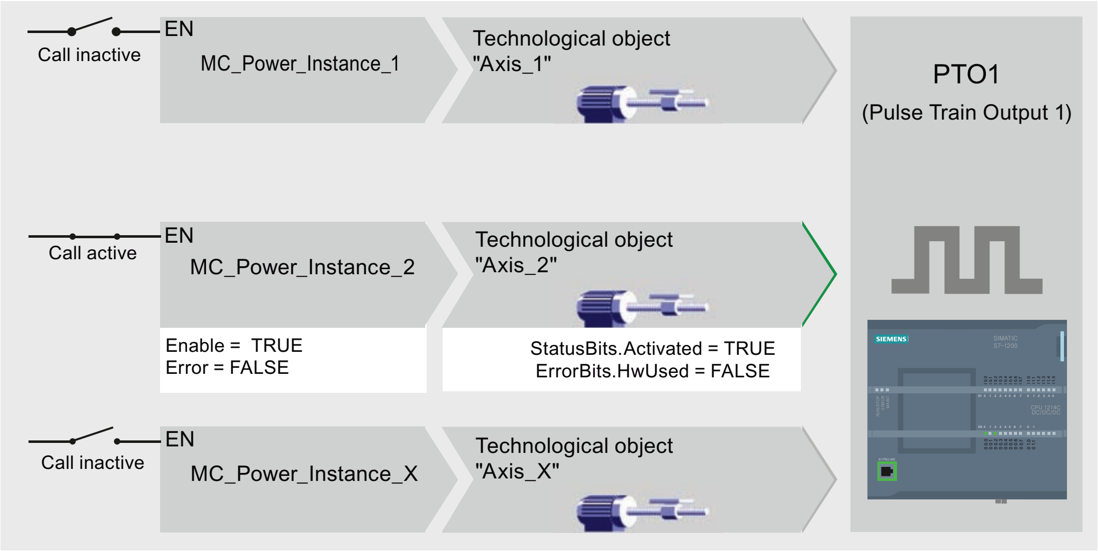


In this example, more than one "Axis" technology object, each with its own axis configuration, uses the same PTO. Each "Axis" must be called in the user program with a separate call of Motion Control instruction "MC_Power" with a separate instance data block. Only one "Axis" at a time may use the PTO. The axis that is currently using the PTO indicates this with tag <Axis name>.StatusBits.Activated = TRUE.

Switching between "Axis" technology objects

The program scheme described below shows you how to switch between different technology objects and, thus, between different axis configurations. To use the same PTO with multiple axes without error indications, only the Motion Control instructions of the axis currently being used may be called.

The following diagram presents this principle using Motion Control instruction "MC_Power" as an example:



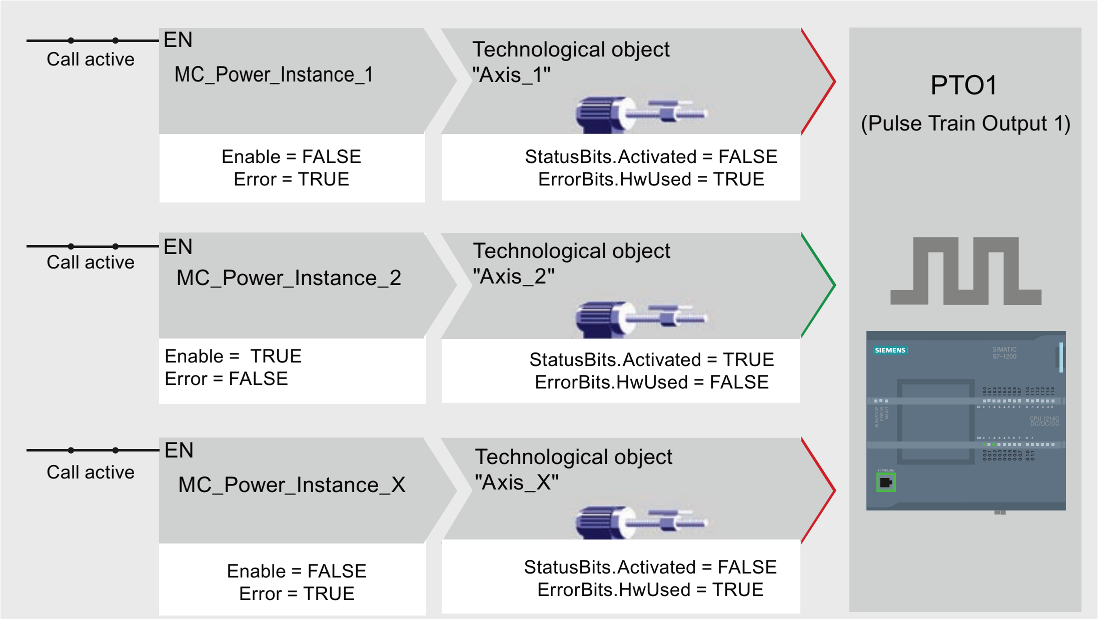
The tags of the activated axis ("Axis_2" here) show the following typical indicators in the user program:

- <Axis name>.StatusBits.Activated = TRUE
- <Axis name>.ErrorBits.HwUsed = FALSE

To switch to the "Axis" technology object, follow the steps described below. In the example, a switch is made from "Axis_2" to "Axis_1":

1. End any active traversing motions of activated "Axis_2"
2. Disable "Axis_2" with the associated Motion Control instruction "MC_Power" using input parameter Enable = FALSE
3. To verify that "Axis_2" has been disabled, use an AND operation of output parameter Status = FALSE of Motion Control instruction "MC_Power" and technology object tag <Axis name>.StatusBits.Enable = FALSE.
4. Deactivate the conditional call of the Motion Control instructions for "Axis_2".
5. Activate the conditional call of the Motion Control instructions for "Axis_1". On the first call of the corresponding Motion Control instruction "MC_Power", "Axis_2" becomes deactivated and "Axis_1" becomes activated.
6. Enable "Axis_1" with Motion Control instruction "MC_Power" using the input parameter Enable = TRUE
7. To verify that "Axis_1" has been enabled, use an AND operation of output parameter Status = TRUE of Motion Control instruction "MC_Power" and technology object tag <Axis name>.StatusBits.Enable = TRUE.

It is also always possible to cyclically call all Motion Control instructions of all axes working with a single PTO.



When an axis is enabled (here "Axis_2"), this axis becomes active.

In contrast to the conditional call, the Motion Control instructions of the deactivated axes (here "Axis_1" and "Axis_x") will indicate errors. The tags of these axes indicate the status <Axis name>.StatusBits.Activated = FALSE and <Axis name>.ErrorBits.HwUsed = TRUE.

Use the conditional call of the Motion Control instructions if you want to implement the user program without error indicators.

See also

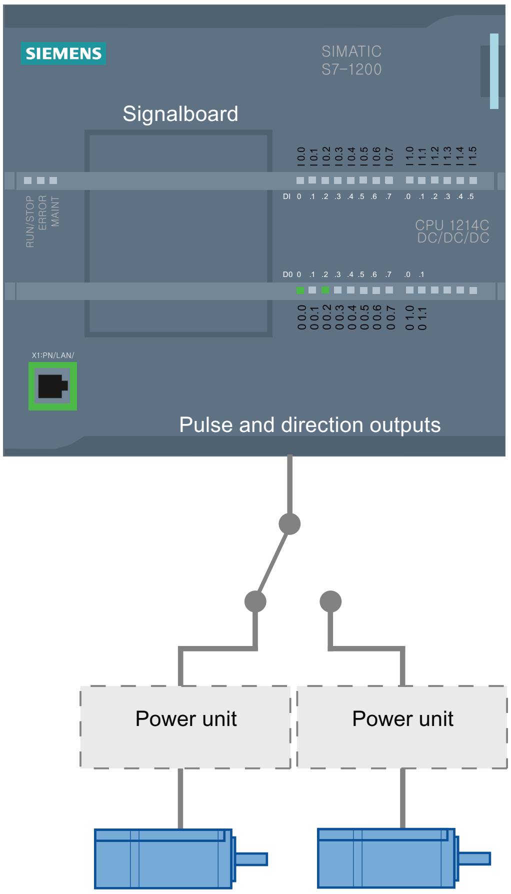
Using multiple drives with the same PTO (Page 3018)

Tracking jobs from higher priority classes (execution levels) (Page 3019)

List of ErrorIDs and ErrorInfos (technology objects as of V2.0) (Page 3028)

Using multiple drives with the same PTO

If multiple drives are to be used, they can be run with a common PTO (Pulse Train Output) using changeover. The following diagram represents the basic circuit design:


SIEMENS
SIMATIC
S7-1200
Signalboard
I 0.0 I 0.1 I 0.2 I 0.3 I 0.4 I 0.5 I 0.6 I 0.7 I 1.0 I 1.1 I 1.2 I 1.3 I 1.4 I 1.5
DI 0 .1 .2 .3 .4 .5 .6 .7 .0 .1 .2 .3 .4 .5
RUN/STOP
ERROR
MAINT
CPU 1214C
DC/DC/DC
D0 0 .1 .2 .3 .4 .5 .6 .7 .0 .1
0 0.0 0 0.1 0 0.2 0 0.3 0 0.4 0 0.5 0 0.6 0 0.7 0 1.0 0 1.1
X1:PN/LAN/
Pulse and direction outputs
Power unit
Power unit

The changeover between drives can be controlled, if required, by the user program via a digital output. If different axis configurations are required for the different drives, a changeover between these configurations is required for the PTO. For additional information on this topic, refer to "Using multiple axes with the same PTO (Page 3015)".

See also

Using multiple axes with the same PTO (Page 3015)

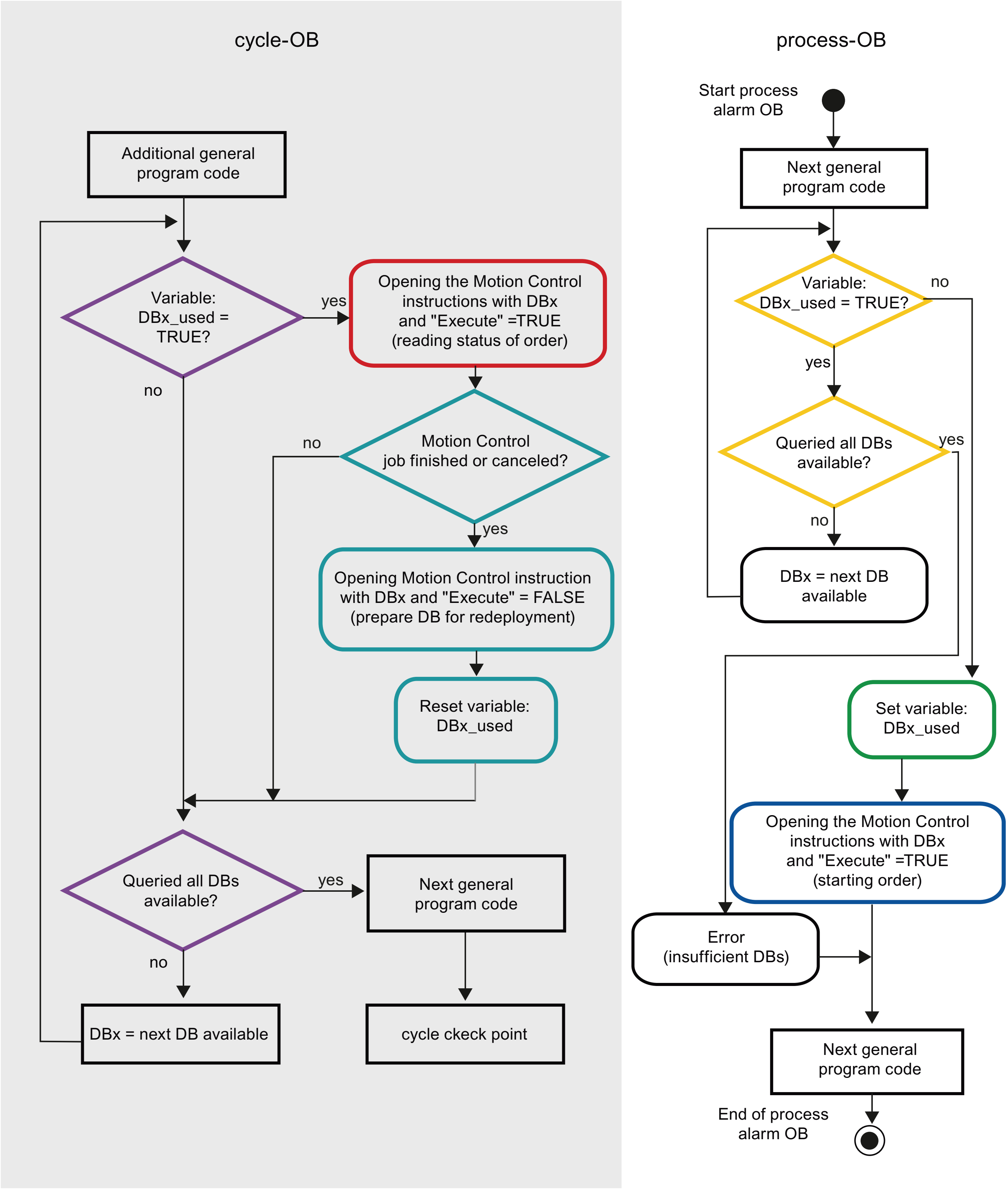
Tracking jobs from higher priority classes (execution levels) (Page 3019)

List of ErrorIDs and ErrorInfos (technology objects as of V2.0) (Page 3028)

Tracking jobs from higher priority classes (execution levels)

Depending on the application, it may be necessary to start motion control jobs (for example, interrupt-controlled) in a higher priority class (execution level).

The Motion Control instructions must be called at short intervals for status monitoring. Motion Control commands cannot be sufficiently closely monitored if the higher priority Motion Control commands are called only once or at too great an interval. Tracking in such cases can be carried out in the cycle OB. An instance data block that is not currently being utilized must be available for each start of a higher priority motion control command. Refer to the following flow chart to see how you start motion control jobs in a higher priority class (for example, hardware interrupt OB) and continue tracking in the program cycle OB:


cycle-OB
Additional general program code
Variable: DBx_used = TRUE?
yes
no
Opening the Motion Control instructions with DBx and "Execute" =TRUE (reading status of order)
Motion Control job finished or canceled?
no
yes
Opening Motion Control instruction with DBx and "Execute" = FALSE (prepare DB for redeployment)
Reset variable: DBx_used
Queried all DBs available?
yes
no
Next general program code
DBx = next DB available
cycle ckeck point
process-OB
Start process alarm OB
Next general program code
Variable: DBx_used = TRUE?
no
yes
Queried all DBs available?
yes
no
DBx = next DB available
Set variable: DBx_used
Opening the Motion Control instructions with DBx and "Execute" =TRUE (starting order)
Error (insufficient DBs)
Next general program code
End of process alarm OB

Depending on the frequency of the motion control jobs you want to start, you will have to generate a sufficient number of instance data blocks. Users determine which instance data block is currently used in the DBx_used tags.

Start of motion control job in the hardware interrupt OB

Binary queries of the DBx_used tags (orange) are used to find an instance data block not currently in use. If such an instance data block is found, the utilized instance data block is marked as "used" (green) and the Motion Control job is started with this instance data block.

Any other program sections of the hardware interrupt OB are then executed, followed by a return to the program cycle OB.

Tracking of started motion control jobs in the program cycle OB

All instance data blocks available in the cycle OB are checked to determine if they are currently in use by means of the DBx_used tag (violet).

If an instance data block is in use (motion control job is being processed), the motion control instruction with this instance data block and input parameter Execute = TRUE is called to read out the status messages (red).

If the job is complete or has been aborted, the following actions are taken next (blue green):

- Call of motion control instruction with input parameter Execute = FALSE
- Resetting the DBx_used tag

This completes the job tracking, and the instance data block is now available for use again.

See also

Using multiple axes with the same PTO (Page 3015)

Using multiple drives with the same PTO (Page 3018)

List of ErrorIDs and ErrorInfos (technology objects as of V2.0) (Page 3028)

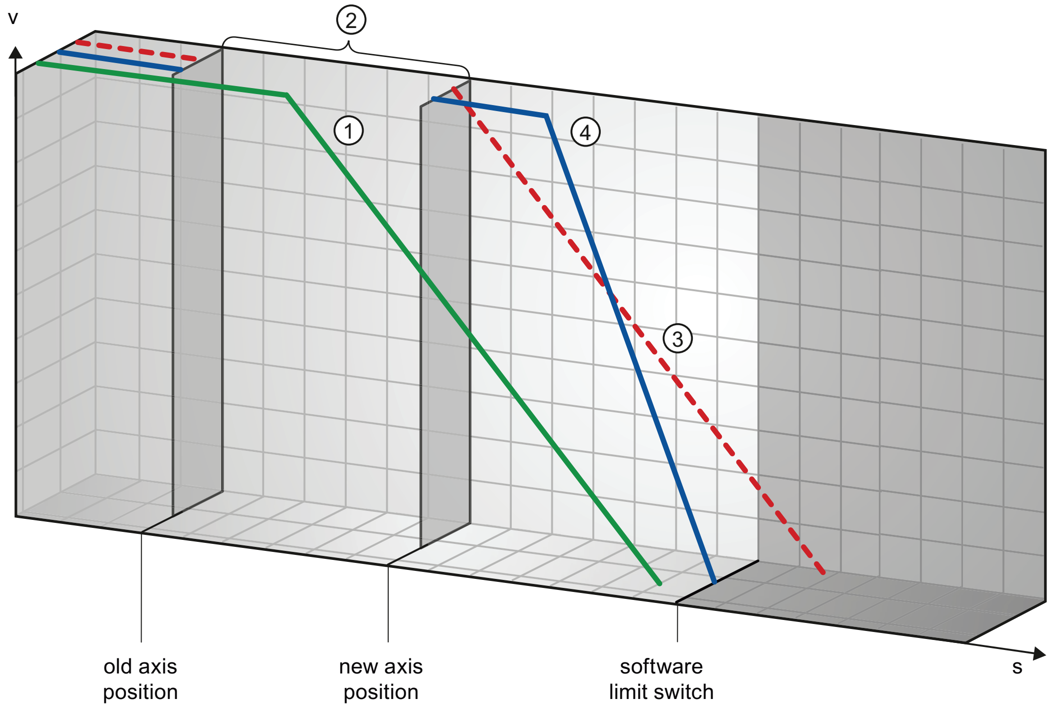
Special cases for use of software limit switches

Software limit switches in conjunction with a homing operation

Due to unfavorably parameterized homing jobs, the braking action of the axis may be influenced at the software limit switch. Take the following examples into consideration when developing your program.

Example 1:

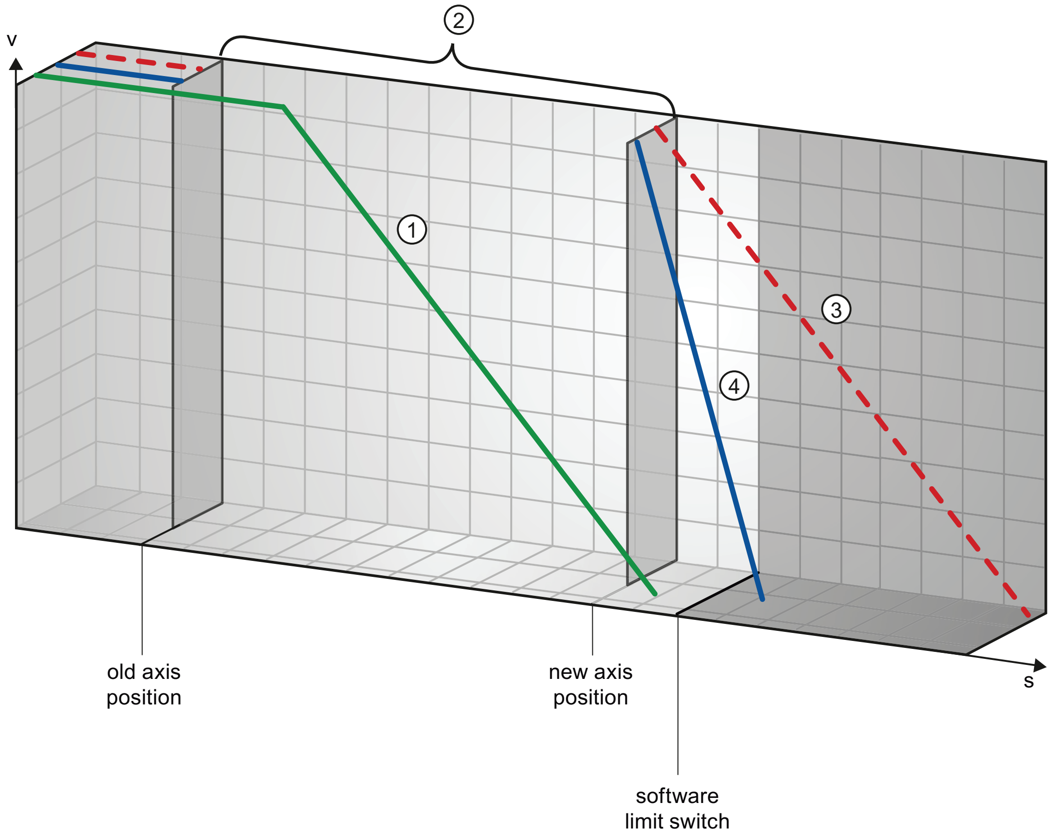
During a travel command, a homing job (for example, Set reference point) offsets the current axis position in the direction of the software limit switch. It is still possible to bring the axis to a standstill before reaching the software limit switch:



①	The green curve shows the motion **without** the homing job. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	A new axis position is set as a result of the homing job. The area between the old and new axis position is thus "skipped".
③	Based on the new axis position, the axis would theoretically be stopped with the configured deceleration at a position after the software limit switch (red curve).
④	Because braking with the configured deceleration is no longer sufficient, the axis actually follows the blue curve. Following a constant motion, the axis brakes at the emergency stop deceleration and comes to a standstill at the position of the software limit switch.

Example 2:

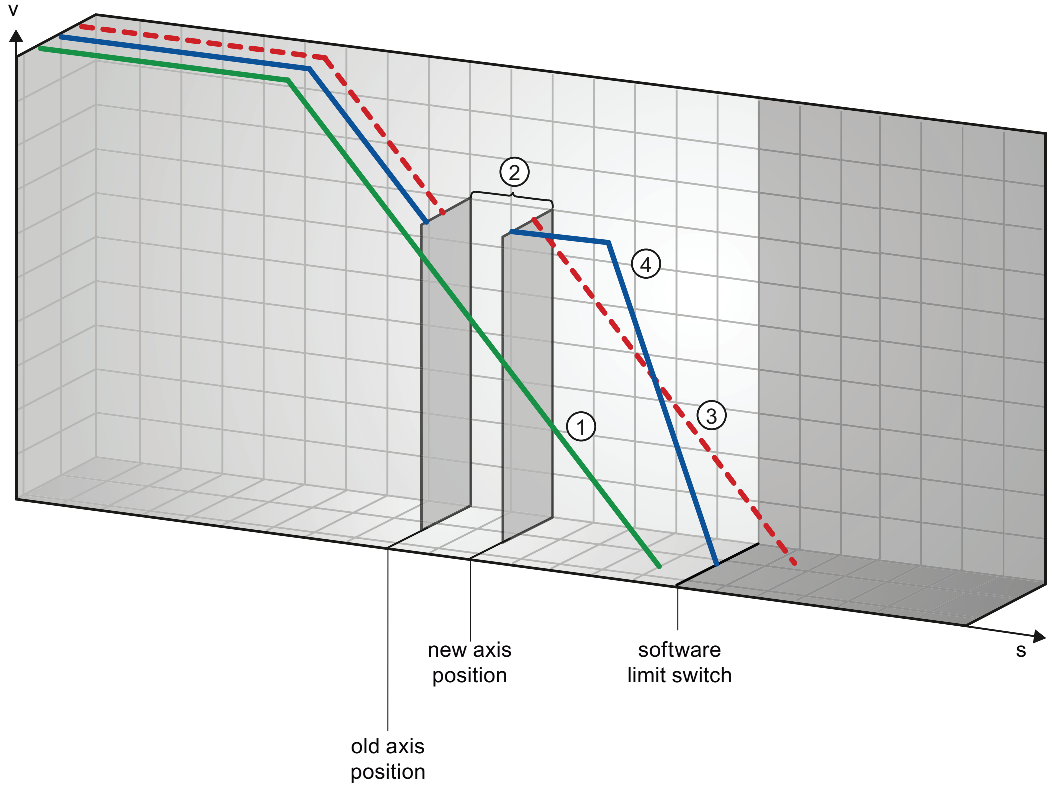
During a travel command, a homing job (for example, Set reference point) offsets the current axis position in the direction of the software limit switch. In contrast to example 1, it is no longer possible to bring the axis to a standstill before reaching the software limit switch. The axis overruns the position of the software limit switch.



①	The green curve shows the motion **without** the homing job. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	A new axis position is set as a result of the homing job. The area between the old and new axis position is thus "skipped".
③	Based on the new axis position, the axis would theoretically be stopped with the configured deceleration at a position well after the software limit switch (red curve).
④	Because braking with the configured deceleration is no longer sufficient, the axis actually follows the blue curve. The axis brakes at the emergency stop deceleration. However, the emergency stop deceleration is not sufficient to stop the axis at the position of the software limit switch. The position of the software limit switch is overrun.

Example 3:

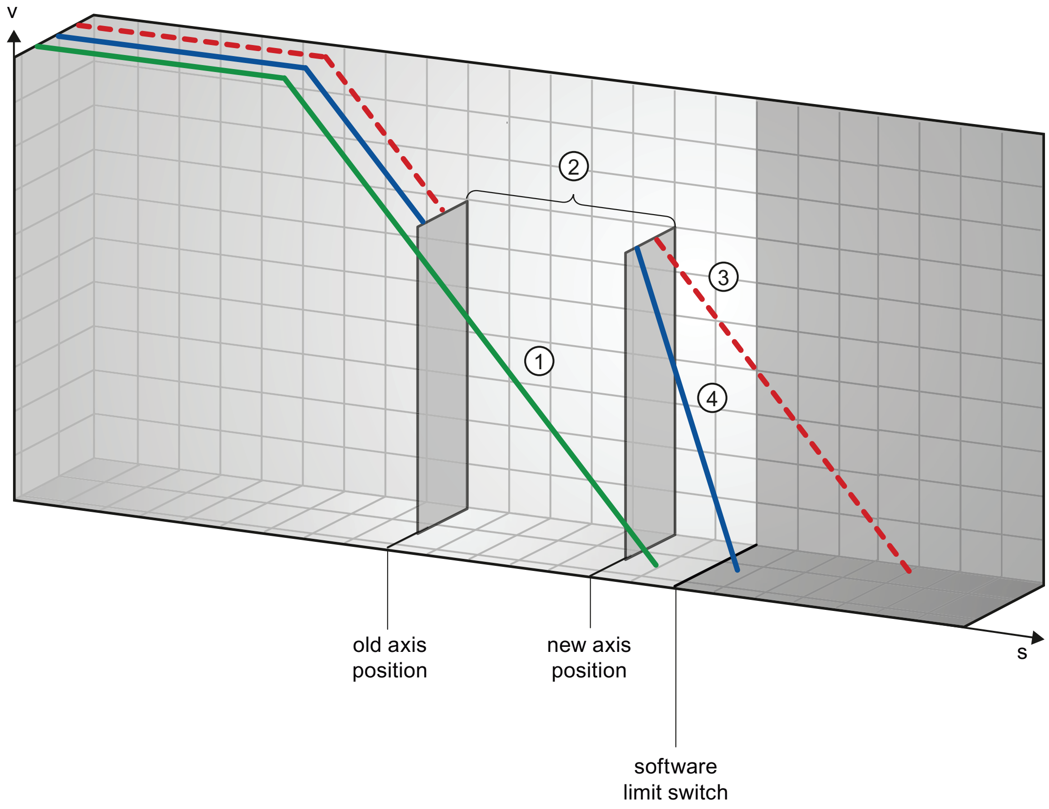
During a braking operation, a homing job (for example, Set reference point) offsets the current axis position in the direction of the software limit switch. It is still possible to bring the axis to a standstill before reaching the software limit switch:



①	The green curve shows the motion **without** the homing job. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	A new axis position is set as a result of the homing job. The area between the old and new axis position is thus "skipped".
③	Based on the new axis position, the axis would theoretically be stopped with the configured deceleration at a position after the software limit switch (red curve).
④	Because braking with the configured deceleration is no longer sufficient, the axis actually follows the blue curve. Following a constant motion, the axis brakes at the emergency stop deceleration and comes to a standstill at the position of the software limit switch.

Example 4:

During a braking operation, a homing job (for example, Set reference point) offsets the current axis position in the direction of the software limit switch. In contrast to example 3, it is no longer possible to bring the axis to a standstill before reaching the software limit switch. The axis overruns the position of the software limit switch.



①	The green curve shows the motion **without** the homing job. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	A new axis position is set as a result of the homing job. The area between the old and new axis position is thus "skipped".
③	Based on the new axis position, the axis would theoretically be stopped with the configured deceleration at a position well after the software limit switch (red curve).
④	Because braking with the configured deceleration is no longer sufficient, the axis actually follows the blue curve. The axis brakes at the emergency stop deceleration. However, the emergency stop deceleration is not sufficient to stop the axis at the position of the software limit switch. The position of the software limit switch is overrun.

See also

Software limit switches and software limit switch position changes. (Page 3026)

Software limit switches in conjunction with dynamic changes (Page 3026)

Behavior of axis when position limits is tripped (Page 2953)

Software limit switches and software limit switch position changes.

An incorrect change in the position of the software limit switch during the runtime of the user program can abruptly reduce the distance between the current axis position and the position of the software limit switch.

The axis response is similar to that described in Software limit switches in conjunction with a homing operation (Page 3021).

See also

Software limit switches in conjunction with a homing operation (Page 3021)

Software limit switches in conjunction with dynamic changes (Page 3026)

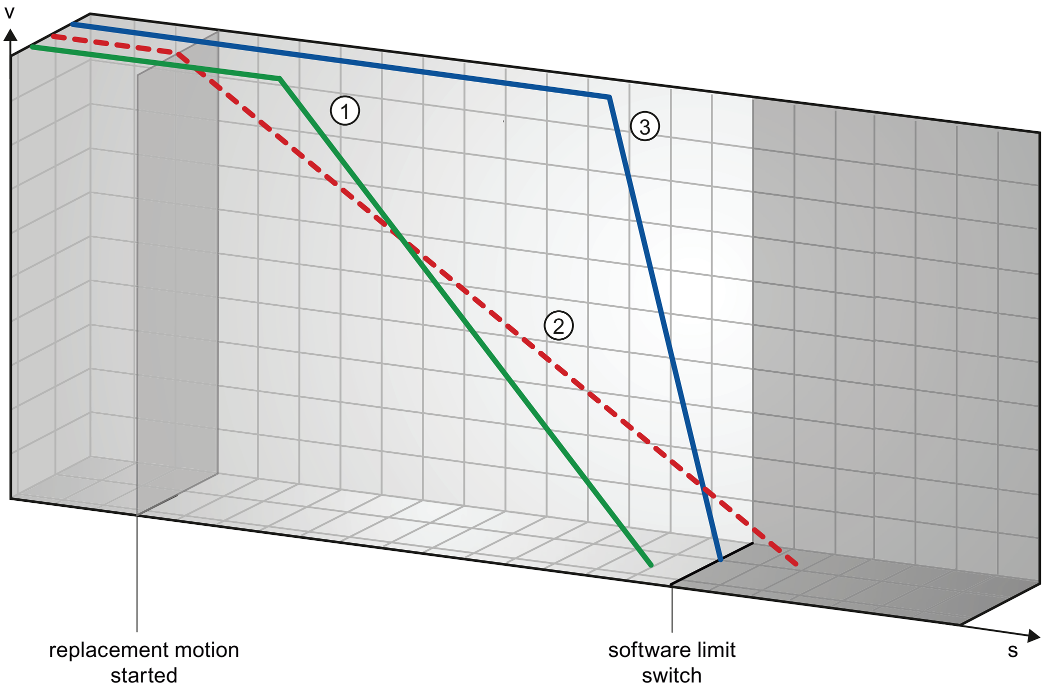
Behavior of axis when position limits is tripped (Page 2953)

Software limit switches in conjunction with dynamic changes

It is possible to influence the deceleration of the axis in the area of the software limit switches in conjunction with overriding motion jobs. This applies when the overriding motion command is started with a lower deceleration (tag <Axis name>.Config.DynamicDefaults.Deceleration). Take the following examples into consideration when developing your program.

Example 1:

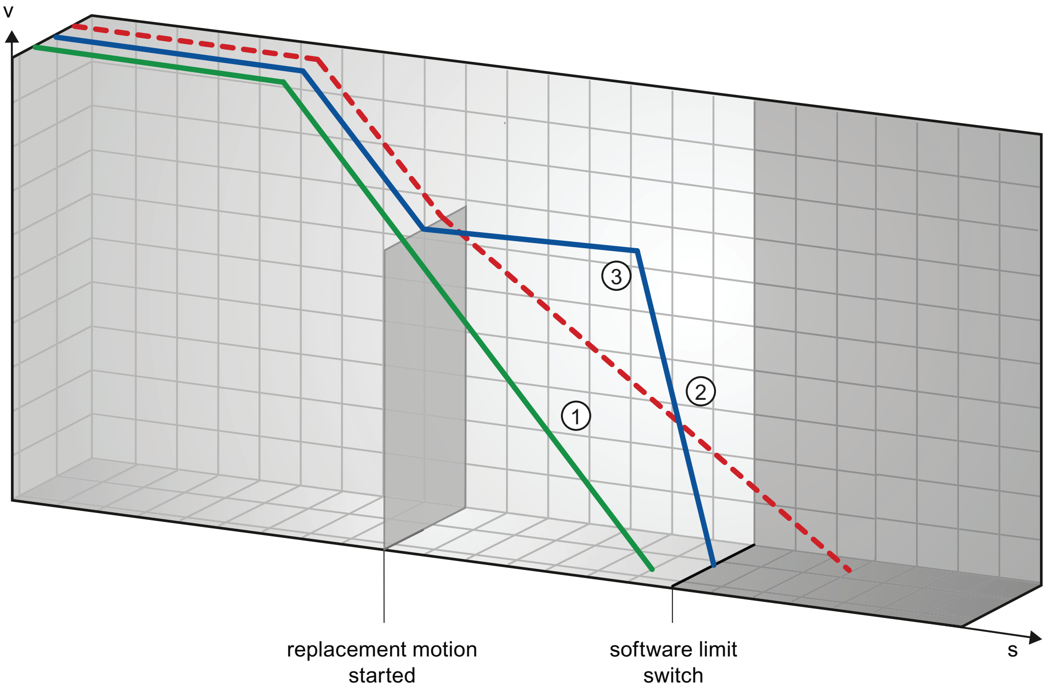
During axis motion, an active motion job is overridden by another motion job with a lower deceleration:



①	The green curve shows the motion of an active job **without** this job being overridden. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	Based on the overriding motion job with lower deceleration, the axis would theoretically be stopped with the configured deceleration at a position after the software limit switch (red curve).
③	Because braking with the configured deceleration of the overriding motion job is no longer sufficient, the axis actually follows the blue curve. Following a constant motion, the axis brakes at the emergency stop deceleration and comes to a standstill at the position of the software limit switch.

Example 2:

During braking of the axis, an active motion job is overridden by another motion job with a lower deceleration:



①	The green curve shows the motion of an active job **without** this job being overridden. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	Based on the overriding motion job with lower deceleration, the axis would theoretically be stopped at a position well after the software limit switch (red curve).
③	Because braking with the configured deceleration of the overriding motion job is no longer sufficient, the axis actually follows the blue curve. Following a constant motion, the axis brakes at the emergency stop deceleration and comes to a standstill at the position of the software limit switch.

See also

Software limit switches in conjunction with a homing operation (Page 3021)

Software limit switches and software limit switch position changes. (Page 3026)

Behavior of axis when position limits is tripped (Page 2953)

Reducing velocity for a short positioning duration

The CPU can reduce the velocity of a positioning command when the planned positioning duration is < 2 ms.

The velocity of command execution will then be reduced for the entire duration. The reduced velocity (pulses per s) is calculated as follows:

- Reduced velocity = Number of pulses to be output * 500Hz

Velocity is not reduced if the planned positioning duration is >= 2 ms.

Dynamic adjustment of start/stop velocity

The configuration of your velocity limits (start/stop velocity and maximum velocity), the dynamic values (acceleration, deceleration and jerk) and the target speed of the traversing command may under certain circumstances result in the start/stop velocity being dynamically adjusted by the CPU.

This is for example the case when, due to a low configured start/stop velocity, the time required for the first pulses would be longer than that possible for the entire acceleration. The first pulse is in these cases output at a greater velocity than the configured start/stop velocity. The subsequent pulses are also dynamically adjusted to ensure the acceleration process can be completed in the specified time.

Ensure in the event of any pulse loss that the hardware (drive) you use is adjusted to this situation, or change the dynamic settings of your axis to avoid dynamic adjustment of the start/stop velocity.

List of ErrorIDs and ErrorInfos (technology objects as of V2.0)

The following table lists all ErrorIDs and ErrorInfos that can be indicated in Motion Control instructions. In addition to the cause of the error, remedies for eliminating the error are also listed:

Operating error with stop of the axis

ErrorID	ErrorInfo	Description	Remedy
16#8000		**Drive error, "Drive ready" failure**	
	16#0001	-	Acknowledge error with instruction "MC_Reset"; provide drive signal; possibly restart command
16#8001		**Low software limit switch has been tripped**	

ErrorID	ErrorInfo	Description	Remedy
	16#000E	The position of the low software limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the software limit switch
	16#000F	The position of the low software limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the low software limit switch was exceeded with the emergency stop deceleration	
16#8002		**High software limit switch has been tripped**	
	16#000E	The position of the high software limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the software limit switch
	16#000F	The position of the high software limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the high software limit switch was exceeded with the emergency stop deceleration	
16#8003		**Low hardware limit switch was approached**	
	16#000E	The low hardware limit switch was approached. The axis was stopped with the emergency stop deceleration. (During an active reference point approach, the homing switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the hardware limit switch.
16#8004		**High hardware limit switch has been approached**	
	16#000E	The high hardware limit switch was approached. The axis was stopped with the emergency stop deceleration. (During an active reference point approach, the homing switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the hardware limit switch.
16#8005		**PTO/HSC are already being used by another axis**	
	16#0001	-	**The axis was configured incorrectly:** Correct the configuration of the PTO (Pulse Train Output) / HSC (High Speed Counter) and download it to the controller
			More than one axis is to run with one PTO: Another axis is using the PTO/HSC. If the current axis is to assume the control, the other axis must be disabled with "MC_Power" Enable = FALSE. (see also Using multiple axes with the same PTO (Page 3015))
16#8006		**A communication error in the control panel has occurred**	
	16#0012	A timeout has occurred	Check the cable connection and press the "Manual control" button again.

Operating error without stop of the axis

ErrorID	ErrorInfo	Description	Remedy
16#8200		**Axis is not enabled**	
	16#0001	-	Enable the axis; restart the command
16#8201		**Axis has already been enabled by another "MC_Power" instance**	
	16#0001	-	Enable the axis with only one "MC_Power" instance
16#8202		**The maximum number of simultaneously active motion control commands has been exceeded (maximum of 200 commands for all motion control technology objects)**	
	16#0001	-	Reduce the number of simultaneously active jobs; restart the command A command is active if parameter "Busy" = TRUE in the Motion Control instruction.
16#8203		**Axis is currently operated in "Manual control" (axis command table)**	
	16#0001	-	Exit "Manual control"; restart the command
16#8204		**Axis is not homed**	
	16#0001	-	Home the axis with instruction "MC_Home"; restart the command
16#8205		**The axis is currently controlled by the user program (the error is only displayed in the axis command table)**	
	16#0013	The axis is enabled in the user program.	Disable axis with instruction "MC_Power" and select "Manual control" again in the axis command table
16#8206		**Technology object not activated yet**	
	16#0001	-	Enable the axis with instruction "MC_Power" Enable = TRUE or enable the axis in the axis command table.
16#8207		**Command rejected**	
	16#0016	Active homing is running; another homing method cannot be started.	Wait for active homing to finish or abort active homing with a motion command, for example, "MC_Halt".
	16#0018	The axis cannot be moved with a command table whilst it is being directly or passively homed.	Wait until direct or passive homing is complete.
	16#0019	The axis cannot be directly or passively homed whilst a command table is being processed.	Wait for command table to finish or abort the command table with a motion command, for example, "MC_Halt".
16#8208		**Difference between maximum and start/stop velocity is invalid**	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	
16#8209		**Invalid acceleration for technology object "Axis"**	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	

Block parameter error

ErrorID	ErrorInfo	Description	Remedy
16#8400		**Invalid value at parameter "Position" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#0005	The value is outside the number range (greater than 1e^{12})	
	16#0006	The value is outside the number range (less than -1e^{12})	
16#8401		**Invalid value at parameter "Distance" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#0005	The value is outside the number range (greater than 1e^{12})	
	16#0006	The value is outside the number range (less than -1e^{12})	
16#8402		**Invalid value at parameter "Velocity" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#0008	Value is greater than the configured maximum velocity	
	16#0009	Value is less than the configured start/stop velocity	
	16#0024	Value is less than 0	
16#8403		**Invalid value at parameter "Direction" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; restart the command
16#8404		**Invalid value at parameter "Mode" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; restart the command
	16#0015	Active/passive homing is not configured	Correct the configuration and download it to the controller; enable the axis and restart the command
	16#0017	The direction reversal is activated at the hardware limit switch, despite the fact that the hardware limit switches are disabled	• Activate the hardware limit switch using the tag <Axis>.Config.PositionLimits_HW.Active = TRUE, restart the command • Correct the configuration and download it to the controller; enable the axis and restart the command
16#8405		**Invalid value at parameter "StopMode" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; enable the axis again
16#8406		**Simultaneous forward and backward jogging is not allowed**	
	16#0001	-	Take steps to ensure that parameters "JogForward" and "JogBackward" do not have signal status TRUE simultaneously; restart the command.
16#8407		**Switching to another axis with instruction "MC_Power" is only permitted after disabling the active axis.**	
	16#0001	-	Disable the active axis; it is then possible to switch to the other axis and enable it.

ErrorID	ErrorInfo	Description	Remedy
16#8408		Invalid value at parameter "Axis" of the Motion Control instruction	
	16#001A	The specified value does not match the required technology object version	Correct the value; restart the command
	16#001B	The specified value does not match the required technology object type	
	16#001C	The specified value is not a Motion Control technology data block	
16#8409		Invalid value at parameter "CommandTable" of the Motion Control instruction	
	16#001A	The specified value does not match the required technology object version	Correct the value; restart the command
	16#001B	The specified value does not match the required technology object type	
	16#001C	The specified value is not a Motion Control technology data block	
16#840A		Invalid value at parameter "StartStep" of the Motion Control instruction	
	16#000A	Value is less than or equal to 0	Correct the value; restart the command
	16#001D	The start step is greater than the end step	
	16#001E	Value is greater than 32	
16#840B		Invalid value at parameter "EndStep" of the Motion Control instruction	
	16#000A	Value is less than or equal to 0	Correct the value; restart the command
	16#001E	Value is greater than 32	
16#840C		Invalid value at parameter "RampUpTime" of the Motion Control instruction	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	
16#840D		Invalid value at parameter "RampDownTime" of the Motion Control instruction	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	
16#840E		Invalid value at parameter "EmergencyRampTime" of the Motion Control instruction	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	
16#840F		Invalid value at parameter "JerkTime" of the Motion Control instruction	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	

Configuration error of the axis

ErrorID	ErrorInfo	Description	Remedy
16#8600		Parameterization of pulse generator (PTO) is invalid	
	16#000B	The address is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0014	The selected hardware is used by another application	
16#8601		Parameterization of the high-speed counter (HSC) is invalid	
	16#000B	The address is invalid	• Download error-free configuration to the

ErrorID	ErrorInfo	Description	Remedy
	16#0014	The selected hardware is used by another application	controller; enable the axis again with instruction "MC_Power"
16#8602		**Invalid parameter assignment of "Enable-Output"**	
	16#000B	The address is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8603		**Invalid parameter assignment of "Ready-Input"**	
	16#000B	The address is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8604		**Invalid "Pulses per motor revolution" value**	
	16#000A	Value is less than or equal to zero	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8605		**Invalid "Load distance per motor revolution" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0005	The value is outside the number range (greater than 1e^{12})	
	16#000A	Value is less than or equal to zero	
16#8606		**Invalid "Start / stop velocity" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0003	Value exceeds the high hardware limit	
	16#0004	Value is less than the low hardware limit	
	16#0007	The start/stop velocity is greater than the maximum velocity	
16#8607		**Invalid "maximum velocity" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0003	Value exceeds the high hardware limit	
	16#0004	Value is less than the low hardware limit	
16#8608		**Invalid "Acceleration" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and if necessary restart the command
	16#0003	Value exceeds the high hardware limit	
	16#0004	Value is less than the low hardware limit	
16#8609		**Invalid "Deceleration" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and if necessary restart the command
	16#0003	Value exceeds the high hardware limit	
	16#0004	Value is less than the low hardware limit	

ErrorID	ErrorInfo	Description	Remedy
16#860A		**Invalid "Emergency stop deceleration" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and if necessary restart the command
	16#0003	Value exceeds the high hardware limit	
	16#0004	Value is less than the low hardware limit	
16#860B		**Value for position of the low SW limit switch is invalid**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
	16#0007	The position value of the low software limit switch is greater than that of the high software limit switch	
16#860C		**Value for position of the high SW limit switch is invalid**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and if necessary restart the command
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#860D		**Invalid address of the low HW limit switch**	
	16#000C	The address of the falling edge is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000D	The address of the rising edge is invalid	
16#860E		**Invalid address of the high HW limit switch**	
	16#000C	The address of the falling edge is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000D	The address of the rising edge is invalid	
16#860F		**Invalid "home position offset" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and if necessary restart the command
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8610		**Invalid "approach velocity" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and if necessary restart the command
	16#0008	The velocity is greater than the maximum velocity	
	16#0009	The velocity is less than the start/stop velocity	

ErrorID	ErrorInfo	Description	Remedy
16#8611		**Invalid "Homing velocity" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and if necessary restart the command
	16#0008	The velocity is greater than the maximum velocity	
	16#0009	The velocity is less than the start/stop velocity	
16#8612		**Invalid address of the homing switch**	
	16#000C	The address of the falling edge is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000D	The address of the rising edge is invalid	
16#8613		**During active homing, direction reversal at the hardware limit switch is activated although the hardware limit switches are not configured**	
	16#0001	-	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and if necessary restart the command
16#8614		**Invalid "Jerk" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and if necessary restart the command
	16#001F	Value is greater than the maximum jerk	
	16#0020	Value is less than the minimum jerk	

Configuration error of the command table

ErrorID	ErrorInfo	Description	Remedy
16#8700		**Value for "Command type" in the command table is invalid**	
	16#0001	-	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and if necessary restart the command
16#8701		**Value for "Position / travel path" in the command table is invalid**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and if necessary restart the command
	16#0005	The value is outside the number range (greater than 1e^{12})	
	16#0006	The value is outside the number range (less than -1e^{12}	
16#8702		**Value for "Velocity" in the command table is invalid**	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and if necessary restart the command
	16#0008	Value is greater than the configured maximum velocity	
	16#0009	Value is less than the configured start/stop velocity	
16#8703		Value for "Duration" in the command table is invalid	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and if necessary restart the command
	16#0021	Value is greater than 64800 s	
	16#0022	Value is less than 0.001 s	
16#8704		Value for "Next step" in the command table is invalid	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and if necessary restart the command
	16#0023	The command transition is not permitted for this command	

Internal errors

ErrorID	ErrorInfo	Description	Remedy
16#8FFF		Internal error	
	16#F0**	-	POWER OFF and POWER ON the CPU If this does not work, contact Customer Support. Have the following information ready: • ErrorID • ErrorInfo • Diagnostic buffer entries

See also

Using multiple axes with the same PTO (Page 3015)

Using multiple drives with the same PTO (Page 3018)

Tracking jobs from higher priority classes (execution levels) (Page 3019)

Tag of the Axis technology object

Config. tag

Config.General. tag

Table 11- 1 Legend

Data type	Data type of the tag	
Start value	Start value of tag The initial value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change in the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.General.PTO				
Tag cannot be evaluated in the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DWORD	DW#16#00000000	-	-	-

<Axis name>.Config.General.HSC				
Tag cannot be evaluated in the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DWORD	DW#16#00000000	-	-	-

<Axis name>.Config.General.LengthUnit ("Axis" technology object as of V2.0)				
The unit of measurement for the parameter selected in the configuration: • 1013 = "mm" • 1010 =: "m" • 1019 = "in" • 1018 = "ft" • 1005 = "°" (degrees) • -1 = "Pulse"				
Data type	Start value	Access	Effective	HMI
Int	1013	R	-	X

Config.DriveInterface. tag

Table 11- 2 **Legend**

<table>
<tr><td><b>Data type</b></td><td colspan="2">Data type of the tag</td></tr>
<tr><td><b>Start value</b></td><td colspan="2">Start value of tag
The initial value can be overwritten by the axis configuration.</td></tr>
<tr><td rowspan="4"><b>Access</b></td><td colspan="2">Access to the tag in the user program:</td></tr>
<tr><td>RW</td><td>The tag can be read and written in the user program.</td></tr>
<tr><td>R</td><td>The tag can be read in the user program.</td></tr>
<tr><td>-</td><td>The tag cannot be used in the user program.</td></tr>
<tr><td><b>Effective</b></td><td colspan="2">Specifies when a change in the tag takes effect.</td></tr>
<tr><td><b>HMI</b></td><td colspan="2">The tag can be used in an HMI system.</td></tr>
</table>

<Axis name>.Config.DriveInterface.EnableOutput...				
Tags cannot be evaluated in the user program.				
Data type	Start value	Access	Effective	HMI
-	-	-	-	-

<Axis name>.Config.DriveInterface.ReadyInput...				
Tags cannot be evaluated in the user program.				
Data type	Start value	Access	Effective	HMI
-	-	-	-	-

Config.Mechanics. tag

Table 11- 3 Legend

Data type	Data type of the tag	
Start value	Start value of tag The initial value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change in the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.Mechanics.PulsesPerDriveRevolution)				
Increments per motor revolution				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DInt	L#1000	R	-	X

<Axis name>.Config.Mechanics.LeadScrew				
Load distance per motor revolution (specified in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	1.0E+001	R	-	X

<Axis name>.Config.Mechanics.InverseDirection				
Invert direction signal				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

Config.DynamicLimits. tag

Table 11- 4 Legend

Data type	Data type of the tag	
Start value	Start value of tag The initial value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program.

	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change in the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.DynamicLimits.MinVelocity				
Start/stop velocity of axis (specified in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	1.0E+001	R	-	X

<Axis name>.Config.DynamicLimits.MaxVelocity				
Maximum velocity of axis (specified in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	2.5E+002	R	-	X

Config.DynamicDefaults. tag

Table 11- 5 **Legend**

Data type	Data type of the tag	
Start value	Start value of tag The initial value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change in the tag takes effect.	
	1	When axis is activated (tag <Axis name>.StatusBits.Activated changes from FALSE -> TRUE), blocked or released
	2	When axis is enabled
	5	The next time a MC_MoveAbsolute-, MC_MoveRelative, MC_MoveVelocity, MC_MoveJog, MC_Halt, MC_CommandTable or active MC_Home-command is started (Mode = 3).
	6	When a MC_MoveJog command is stopped
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.DynamicDefaults.Acceleration					
Acceleration of axis (specified in the configured dimension unit)					
Data type	Start value	Access	Effective		HMI
Real	4.8E+001	RW	5	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

<Axis name>.Config.DynamicDefaults.Deceleration					
Deceleration of axis (specified in the configured dimension unit)					
Data type	Start value	Access	Effective		HMI
Real	4.8E+001	RW	5, 6	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

<Axis name>.Config.DynamicDefaults.EmergencyDeceleration					
Emergency stop deceleration of axis (specified in the configured dimension unit)					
Data type	Start value	Access	Effective		HMI
Real	1.2E+002	RW	2, 5, 6	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

<Axis name>.Config.DynamicDefaults.JerkActive ("Axis" technology object as of V2.0)				
TRUE = the jerk limit is activated				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	RW	1, 5	X

<Axis name>.Config.DynamicDefaults.Jerk ("Axis" technology object as of V2.0)				
Jerk during axis acceleration and deceleration ramp (specified in the configured unit of measurement)				
Data type	Start value	Access	Effective	HMI
Real	1.92E+002	RW	1, 5	X

Config.PositionLimits_SW. tag

Table 11- 6 **Legend**

Data type	Data type of the tag	
Start value	Start value of tag The initial value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change in the tag takes effect.	
	1	When axis is activated (tag <Axis name>.StatusBits.Activated changes from FALSE -> TRUE), blocked or released
	4	Upon the next start of a Motion Control command after a standstill of the axis. The axis standstill can be checked with tag <Axis name>. StatusBits.Standstill.
	5	The next time a MC_MoveAbsolute, MC_MoveRelative-, MC_MoveVelocity, MC_MoveJog, MC_Halt, MC_CommandTable or active MC_Homecommand is started (Mode = 3).
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.PositionLimits_SW.Active					
TRUE = The software limit switches are activated					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Bool	FALSE	RW	4	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

<Axis name>.Config.PositionLimits_SW.MinPosition					
Position of low software limit switch (specified in the configured unit of measurement)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Real	-1.0E+004	RW	4	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

<Axis name>.Config.PositionLimits_SW.MaxPosition					
Position of high software limit switch (specified in the configured unit of measurement)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Real	1.0E+004	RW	4	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

Config.PositionLimits_HW. tag

Table 11- 7 Legend

Data type	Data type of the tag	
Start value	Start value of tag The initial value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change in the tag takes effect.	
	1	When axis is activated (tag <Axis name>.StatusBits.Activated changes from FALSE -> TRUE), blocked or released
	3	After axis enable (the axis must have previously been at a standstill). The axis standstill can be checked with tag <Axis name>. StatusBits.Standstill.
	4	Upon the next start of a Motion Control command after a standstill of the axis. The axis standstill can be checked with tag <Axis name>. StatusBits.Standstill.
	5	The next time aMC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity, MC_MoveJog, MC_Halt, MC_CommandTable or active MC_Home command is started (Mode = 3).
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.PositionLimits_HW.Active					
TRUE = The hardware limit switches are active.					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Bool	FALSE	RW	3, 4	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

<table>
<tr><th colspan="5">&lt;Axis name>.Config.PositionLimits_HW.MinSwitchedLevel</th></tr>
<tr><td colspan="5">TRUE = 24 V at CPU input corresponds to low hardware limit switch approached
FALSE = 0 V at CPU input corresponds to low hardware limit switch approached</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td>Effective</td><td>HMI</td></tr>
<tr><td>Bool</td><td>FALSE</td><td>R</td><td>-</td><td>X</td></tr>
</table>

<table>
<tr><th colspan="5">&lt;Axis name>.Config.PositionLimits_HW.MinFallingEvent</th></tr>
<tr><td colspan="5">Tag cannot be evaluated in the user program.</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td>Effective</td><td>HMI</td></tr>
<tr><td>DWord</td><td>DW#16#00000000</td><td>-</td><td>-</td><td>-</td></tr>
</table>

<table>
<tr><th colspan="5">&lt;Axis name>.Config.PositionLimits_HW.MinRisingEvent</th></tr>
<tr><td colspan="5">Tag cannot be evaluated in the user program.</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td>Effective</td><td>HMI</td></tr>
<tr><td>DWord</td><td>DW#16#00000000</td><td>-</td><td>-</td><td>-</td></tr>
</table>

<table>
<tr><th colspan="5">&lt;Axis name>.Config.PositionLimits_HW.MaxSwitchedLevel</th></tr>
<tr><td colspan="5">TRUE = 24 V at CPU input corresponds to high hardware limit switch approached
FALSE = 0 V at CPU input corresponds to high hardware limit switch approached</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td>Effective</td><td>HMI</td></tr>
<tr><td>Bool</td><td>FALSE</td><td>R</td><td>-</td><td>X</td></tr>
</table>

<table>
<tr><th colspan="5">&lt;Axis name>.Config.PositionLimits_HW.MaxFallingEvent</th></tr>
<tr><td colspan="5">Tag cannot be evaluated in the user program.</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td>Effective</td><td>HMI</td></tr>
<tr><td>DWord</td><td>DW#16#00000000</td><td>-</td><td>-</td><td>-</td></tr>
</table>

<table>
<tr><th colspan="5">&lt;Axis name>.Config.PositionLimits_HW.MaxRisingEvent</th></tr>
<tr><td colspan="5">Tag cannot be evaluated in the user program.</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td>Effective</td><td>HMI</td></tr>
<tr><td>DWord</td><td>DW#16#00000000</td><td>-</td><td>-</td><td>-</td></tr>
</table>

Config.Homing. tag

Table 11- 8 Legend

Data type	Data type of the tag	
Start value	Start value of tag The initial value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change in the tag takes effect.	
	1	When axis is activated (tag <Axis name>.StatusBits.Activated changes from FALSE -> TRUE), blocked or released
	7	When a passive homing command is started
	8	When an active homing command is started
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.Homing.AutoReversal					
TRUE = Direction reversal at hardware limit switch enabled (active homing) FALSE = Direction reversal at hardware limit switch disabled (active homing)					
Data type	**Start value**	**Access**	**effective**		**HMI**
Bool	TRUE	R	-	Technology object "Axis" V1.0	X
		RW	1, 8	Technology object "Axis" V2.0	

<Axis name>.Config.Homing.Direction					
TRUE = Positive approach direction to search for homing switch and positive homing direction (active homing) FALSE = Negative approach direction to search for homing switch and positive homing direction (active homing)					
Data type	**Start value**	**Access**	**effective**		**HMI**
Bool	TRUE	R	-	Technology object "Axis" V1.0	X
		RW	1, 8	Technology object "Axis" V2.0	

<table>
<tr><th colspan="5">&lt;Axis name>.Config.Homing.SideActiveHoming ("Axis" technology object as of V2.0)</th></tr>
<tr><td colspan="5">TRUE = Homing on high side of the homing switch (active homing)
TRUE = Homing on lower side of the homing switch (active homing)</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td>Effective</td><td>HMI</td></tr>
<tr><td>Bool</td><td>TRUE</td><td>RW</td><td>1, 8</td><td>X</td></tr>
</table>

<table>
<tr><th colspan="5">&lt;Axis name>.Config.Homing.SidePassiveHoming ("Axis" technology object as of V2.0)</th></tr>
<tr><td colspan="5">TRUE = Homing on high side of the homing switch (passive homing)
TRUE = Homing on lower side of the homing switch (passive homing)</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td>Effective</td><td>HMI</td></tr>
<tr><td>Bool</td><td>TRUE</td><td>RW</td><td>1, 7</td><td>X</td></tr>
</table>

<table>
<tr><th colspan="5">&lt;Axis name>.Config.Homing.RisingEdge (as of technology object "Axis" V1.0)</th></tr>
<tr><td colspan="5">TRUE = Homing with negative signal edge of the homing switch (active homing)
FALSE = Homing with positive signal edge of the homing switch (active homing)
For information on the effect of the tag on passive homing, refer to the description in "Configuration - Homing".</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td>Effective</td><td>HMI</td></tr>
<tr><td>Bool</td><td>FALSE</td><td>R</td><td>-</td><td>X</td></tr>
</table>

<table>
<tr><th colspan="6">&lt;Axis name>.Config.Homing.Offset</th></tr>
<tr><td colspan="6">Home position offset /specified in the configured unit of measurement (active homing)</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td colspan="2">Effective</td><td>HMI</td></tr>
<tr><td rowspan="2">Real</td><td rowspan="2">0.0</td><td rowspan="2">R
RW</td><td>-</td><td>Technology object "Axis" V1.0</td><td rowspan="2">X</td></tr>
<tr><td>1, 8</td><td>Technology object "Axis" V2.0</td></tr>
</table>

<table>
<tr><th colspan="6">&lt;Axis name>.Config.Homing.FastVelocity</th></tr>
<tr><td colspan="6">Approach velocity / specified in the configured unit of measurement (active homing)</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td colspan="2">Effective</td><td>HMI</td></tr>
<tr><td rowspan="2">Real</td><td rowspan="2">2.0E+002</td><td>R</td><td>-</td><td>Technology object "Axis" V1.0</td><td rowspan="2">X</td></tr>
<tr><td>RW</td><td>1, 8</td><td>Technology object "Axis" V2.0</td></tr>
</table>

<Axis name>.Config.Homing.SlowVelocity					
Homing velocity / specified in the configured unit of measurement (active homing)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Real	4.0E+001	R	-	Technology object "Axis" V1.0	X
		RW	1, 8	Technology object "Axis" V2.0	

<Axis name>.Config.Homing.FallingEvent				
Tag cannot be evaluated in the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DWord	DW#16#00000000	-	-	-

<Axis name>.Config.Homing.RisingEvent				
Tag cannot be evaluated in the user program.				
Data type	**Start value**	**Access**	**effective**	**HMI**
DWord	DW#16#00000000	-	-	

MotionStatus. tag

Table 11- 9 **Legend**

Data type	Data type of the tag	
Start value	Start value of tag	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change in the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.MotionStatus.Position				
Current position of the axis (specified in the configured unit of measurement) If the axis is not homed, the tag indicates the position value relative to the enable position of the axis.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	R	-	X

<Axis name>..MotionStatus.Velocity				
Current velocity of the axis (specified in the configured unit of measurement)				
Data type	Start value	Access	Effective	HMI
Real	0.0	R	-	X

<Axis name>.MotionStatus.Distance				
Current distance to the target position of the axis (specified in the configured unit of measurement) The value of the tag is only valid during execution of a positioning command with "MC_MoveAbsolute" or "MC_MoveRelative" or of the axis command table.				
Data type	Start value	Access	Effective	HMI
Real	0.0	R	-	X

<Axis name>.MotionStatus.TargetPosition				
Target position of axis (specified in the configured unit of measurement) The value of the tag is only valid during execution of a positioning command with "MC_MoveAbsolute" or "MC_MoveRelative" or of the axis command table.				
Data type	Start value	Access	Effective	HMI
Real	0.0	R	-	X

See also

Motion status (Page 3013)

StatusBits. tag

Table 11- 10 **Legend**

Data type	Data type of the tag	
Start value	Start value of tag	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change in the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.StatusBits.Activated				
TRUE = The axis is activated. It is connected to the assigned PTO (Pulse Train Output). The data of the technology data block will be updated cyclically.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Enable				
TRUE = The axis is enabled and ready to take on Motion Control jobs.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.HomingDone				
TRUE = The axis is homed and is capable of executing absolute positioning jobs. The axis does not have to be homed for relative homing. The status is FALSE during active homing. The status will remain TRUE during passive homing if the axis has already been homed.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Done				
TRUE = No Motion Control command is active on the axis.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Error				
TRUE = An error occurred in the axis technology object. Detailed information about the error is available in automatic mode in the ErrorID" and ErrorInfo" parameters of the motion control instructions. In manual mode, the "Error message" field of the axis command table displays detailed information about the cause of error.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.StandStill				
TRUE = The axis is at a standstill.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.PositioningCommand				
TRUE = The axis is executing a positioning command.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.StatusBits.SpeedCommand				
TRUE = The axis is executing a travel command at predefined velocity.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Homing				
TRUE = The axis is executing a homing command of the "MC_Home" Motion Control instruction or axis command table.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.StatusBits.CommandTableActive				
TRUE = The axis is controlled by Motion Control instruction "MC_CommandTable".				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.StatusBits.ConstantVelocity				
TRUE = The axis travels at constant velocity.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Acceleration				
TRUE = The axis accelerates.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Deceleration				
TRUE = The axis decelerates (slows down).				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.StatusBits.ControlPanelActive				
TRUE = The "Manual control" mode has been enabled in the axis command table. The axis command table has control priority over the "Axis" technology object. The axis cannot be controlled from the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.DriveReady				
TRUE = The drive is ready.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

See also

Status and error bits (Page 3010)

ErrorBits. tag

Table 11- 11 Legend

Data type	Data type of the tag	
Start value	Start value of tag	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change in the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.ErrorBits.SystemFault				
TRUE = Internal system error.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.ConfigFault				
TRUE = Incorrect configuration of axis.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<table>
<tr><th colspan="5"><Axis name>.ErrorBits.DriveFault</th></tr>
<tr><td colspan="5">TRUE = The drive has reported an error after failure of its "Drive ready" signal.</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td>Effective</td><td>HMI</td></tr>
<tr><td>Bool</td><td>FALSE</td><td>R</td><td>-</td><td>X</td></tr>
</table>

<table>
<tr><th colspan="5"><Axis name>.ErrorBits.SwLimitMinReached</th></tr>
<tr><td colspan="5">TRUE = The low software limit switch has been reached.</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td>Effective</td><td>HMI</td></tr>
<tr><td>Bool</td><td>FALSE</td><td>R</td><td>-</td><td>X</td></tr>
</table>

<table>
<tr><th colspan="5"><Axis name>.ErrorBits.SwLimitMinExceeded</th></tr>
<tr><td colspan="5">TRUE = The low software limit switch has been exceeded.</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td>Effective</td><td>HMI</td></tr>
<tr><td>Bool</td><td>FALSE</td><td>R</td><td>-</td><td>X</td></tr>
</table>

<table>
<tr><th colspan="5"><Axis name>.ErrorBits.SwLimitMaxReached</th></tr>
<tr><td colspan="5">TRUE = The high software limit switch has been reached.</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td>Effective</td><td>HMI</td></tr>
<tr><td>Bool</td><td>FALSE</td><td>R</td><td>-</td><td>X</td></tr>
</table>

<table>
<tr><th colspan="5"><Axis name>.ErrorBits.SwLimitMaxExceeded</th></tr>
<tr><td colspan="5">TRUE = The high software limit switch has been exceeded.</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td>Effective</td><td>HMI</td></tr>
<tr><td>Bool</td><td>FALSE</td><td>R</td><td>-</td><td>X</td></tr>
</table>

<table>
<tr><th colspan="5"><Axis name>.ErrorBits.HwLimitMin</th></tr>
<tr><td colspan="5">TRUE = The low hardware limit switch has been approached.</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td>Effective</td><td>HMI</td></tr>
<tr><td>Bool</td><td>FALSE</td><td>R</td><td>-</td><td>X</td></tr>
</table>

<table>
<tr><th colspan="5"><Axis name>.ErrorBits.HwLimitMax</th></tr>
<tr><td colspan="5">TRUE = The high hardware limit switch has been approached.</td></tr>
<tr><td>Data type</td><td>Start value</td><td>Access</td><td>Effective</td><td>HMI</td></tr>
<tr><td>Bool</td><td>FALSE</td><td>R</td><td>-</td><td>X</td></tr>
</table>

<Axis name>.ErrorBits.HwUsed				
TRUE = A second axis is using the same PTO (Pulse Train Output) and is enabled with "MC_Power".				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

See also

Status and error bits (Page 3010)

Internal. tag

The "Internal" tags contain no user-relevant data; these tags cannot be accessed in the user program.

ControlPanel tag

The "ControlPanel" tags contain no user-relevant data; these tags cannot be accessed in the user program.

Update of the technology object tags

The status and error information of the axis indicated in the technology object tags is updated at each cycle control point.

The change in values of editable configuration tags does not take effect immediately. For information on the conditions under which a change takes effect, refer to the detailed description of the relevant tag.

Command table technology object tag

Config.Command.Command[1 ... 32] tag

Table 11- 12 **Legend**

Data type	Data type of the tag	
Start value	Start value of tag The start value can be overwritten by the configuration of the command table.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change in the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.Command.Command[x].Type				
Command type • 0 = "Empty" command • 2 = "Hold" command • 5 = "Relative positioning" command • 6 = "Absolute positioning" command • 7 = "Velocity setpoint" command • 151 = "Wait" command				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Int	0	RW	-	X

<Axis name>. Config.Command.Command[x].Position				
Command target position / travel path				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real		RW	-	X

<Axis name>. Config.Command.Command[x].Velocity				
Command velocity				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RW	-	X

<Axis name>. Config.Command.Command[x].Duration				
Command duration				
Data type	Start value	Access	Effective	HMI
Real	0.0	RW	-	X

<Axis name>. Config.Command.Command[x].BufferMode				
Value for command "Next step" • 0 = "Complete command" • 1 = "Blend movement"				
Data type	Start value	Access	Effective	HMI
Int	0	RW	-	X

<Axis name>. Config.Command.Command[x].StepCode				
Command step code				
Data type	Start value	Access	Effective	HMI
Word	0	RW	-	X

Documentation for functions from previous versions

Configuration - Homing (technology object "Axis" V1.0)

Configure the parameters for active and passive homing in the "Homing" configuration window. The homing method is set using the "Mode" input parameter of the motion control instruction. Here, Mode = 2 means passive homing and Mode = 3 means active homing.

Homing switch input

Select the digital input for the homing switch from the drop-down list. The input must be interrupt-capable. The onboard CPU inputs and the inputs of an inserted signal board can be selected as inputs for the homing switch.

Note

The digital inputs are set to a filter time of 6.4 ms by default.

When the digital inputs are used as a homing switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the homing switch, the reference point may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

The specified filter time must be less than the duration of the input signal at the homing switch.

Permitting direction reversal after reaching the HW limit switch (active homing only)

Activate the check box to use the hardware limit switch as a reversing cam for the reference point approach. The hardware limit switches must be activated for direction reversal. If the CPU firmware V1.0 is used, both hardware limit switches must be configured. If CPU firmware as of V2.0 is used, only the hardware limit switches in the approach direction must be configured.

If the hardware limit switch is reached during active homing, the axis brakes at the configured deceleration (not with the emergency stop deceleration) and reverses direction. The homing switch is then sensed in reverse direction.

If the direction reversal is not active and the axis reaches the hardware limit switch during active homing, the reference point approach is aborted with an error and the axis is braked at the emergency stop deceleration.

NOTICE

Use one of the following measures to ensure that the machine does not travel to a mechanical endstop in the event of a direction reversal:

- Keep the approach velocity low
- Increase the configured acceleration/deceleration
- Increase the distance between hardware limit switch and mechanical stop

Approach / homing direction (active and passive homing)

With the direction selection, you determine the "approach direction" used during active homing to search for the homing switch, as well as the homing direction. The homing direction specifies the travel direction the axis uses to approach the configured side of the homing switch to carry out the homing operation.

Refer to the table under "Homing switches" for the effect of the approach direction setting on passive homing.

Side of the homing switch (active and passive homing)

- Active homing

  This is where you select whether the axis is homed on the low or high side of the homing switch.

  Note

  Depending on the start position of the axis and the configuration of the homing parameters, the reference point approach sequence can differ from the diagram in the configuration window.

- Passive homing

  With passive homing, the traversing motions for purposes of homing must be implemented by the user via motion commands. The side of the homing switch on which homing occurs depends on the following factors:

  – "Approach direction" configuration

  – "Homing switch" configuration

  – Current travel direction during passive homing

  The table below presents details on the effect of factors:

Influencing factors:			Result:
Configuration Approach direction	Configuration Homing switch	Current travel direction	Homing on Homing switch
Positive	"Bottom side"	Positive direction	Top side
		Negative direction	Bottom side
Positive	"Top side"	Positive direction	Bottom side
		Negative direction	Top side
Negative	"Bottom side"	Positive direction	Bottom side
		Negative direction	Top side
Negative	"Top side"	Positive direction	Top side
		Negative direction	Bottom side

Velocity (active homing only)

In this field, specify the velocity at which the homing switch is to be searched for during the reference point approach.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ approach velocity ≤ maximum velocity

Homing velocity (active homing only)

In this field, specify the velocity at which the axis approaches the homing switch for homing.

Limits (independent of the selected unit of measurement):

- Start/stop speed ≤ Homing velocity ≤ Maximum velocity

Home position offset (active homing only)

If the desired reference position deviates from the position of the homing switch, the home position offset can be specified in this field.

If the value does not equal 0, the axis executes the following actions following homing at the homing switch:

1. Move the axis at the homing velocity by the value of the home position offset
2. Upon reaching the "home position offset", the axis is at the home position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

Limits (independent of the selected unit of measurement):

- -1.0e12 ≤ home position offset ≤ 1.0e12

Home position

The position for which parameters are assigned in the Motion Control instruction "MC_Home" is used as the home position.

List of ErrorIDs and ErrorInfos (technology objects V1.0)

The following table lists all ErrorIDs and ErrorInfos that can be indicated in motion control instructions. In addition to the cause of the error, remedies for eliminating the error are also listed:

Operating error with stop of the axis

ErrorID	ErrorInfo	Description	Remedy
16#8000		**Drive error, "Drive ready" failure**	
	16#0001	-	Acknowledge error with instruction "MC_Reset"; provide drive signal; possibly restart command
16#8001		**Low software limit switch has been tripped**	
	16#000E	The position of the low software limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the software limit switch
	16#000F	The position of the low software limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the low software limit switch was exceeded with the emergency stop deceleration	

ErrorID	ErrorInfo	Description	Remedy
16#8002		High software limit switch has been tripped	
	16#000E	The position of the high software limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the software limit switch
	16#000F	The position of the high software limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the high software limit switch was exceeded with the emergency stop deceleration	
16#8003		Low hardware limit switch was approached	
	16#000E	The low hardware limit switch was approached. The axis was stopped with the emergency stop deceleration. (During an active reference point approach, the homing switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the hardware limit switch.
16#8004		High hardware limit switch was approached	
	16#000E	The high hardware limit switch was approached. The axis was stopped with the emergency stop deceleration. (During an active reference point approach, the homing switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the hardware limit switch.
16#8005		PTO/HSC are already being used by another axis	
	16#0001	-	The axis was configured incorrectly: Correct the configuration of the PTO (Pulse Train Output) / HSC (High Speed Counter) and download it to the controller
			More than one axis is to run with one PTO: Another axis is using the PTO/HSC. If the current axis is to assume the control, the other axis must be disabled with "MC_Power" Enable = FALSE. (see also Using multiple axes with the same PTO (Page 3015))

Operating error without stop of the axis

ErrorID	ErrorInfo	Description	Remedy
16#8200		Axis is not enabled	
	16#0001	-	Enable the axis; restart the command
16#8201		Axis has already been enabled by another "MC_Power" instance	
	16#0001	-	Enable the axis with only one "MC_Power" instruction
16#8202		The maximum number of simultaneously active motion control jobs was exceeded (maximum of 200 jobs for all motion control technology objects)	

ErrorID	ErrorInfo	Description	Remedy
	16#0001	-	Reduce the number of simultaneously active jobs; restart the command A command is active if parameter "Busy" = TRUE in the motion control instruction.
16#8203		**Axis is currently operated in "Manual control" (axis command table)**	
	16#0001	-	Exit "Manual control"; restart the command
16#8204		**Axis is not homed**	
	16#0001	-	Home the axis with instruction "MC_Home"; restart the command
16#8205		**The axis is currently controlled by the user program (the error is only displayed in the axis command table)**	
	16#0001	-	Disable axis with instruction "MC_Power" and select "Manual control" again in the axis command table
16#8206		**Technology object Axis not yet enabled**	
	16#0001	-	Enable the axis with instruction "MC_Power" Enable = TRUE or enable the axis in the axis command table.
16#8207		**Command rejected**	
	16#0016	Active homing is running; another homing method cannot be started.	Wait for active homing to finish or abort the active homing with a motion command, for example, "MC_Halt". The other homing type can then be started.

Block parameter error

ErrorID	ErrorInfo	Description	Remedy
16#8400		**Invalid value at parameter "Position" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the "position" value; restart the command
	16#0005	The value is outside the number range (greater than 1e^{12})	
	16#0006	The value is outside the number range (less than -1e^{12})	
16#8401		**Invalid value at parameter "Distance" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the "Distance" value; restart the command
	16#0005	The value is outside the number range (greater than 1e^{12})	
	16#0006	The value is outside the number range (less than -1e^{12})	
16#8402		**Invalid value at parameter "Velocity" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the "Velocity" value; restart the command
	16#0008	Velocity is greater than the maximum velocity	
	16#0009	Velocity is less than the start/stop velocity	
16#8403		**Invalid value at parameter "Direction" of the Motion Control instruction**	
	16#0011	Invalid selection value	Correct the selection value; restart the command

ErrorID	ErrorInfo	Description	Remedy
16#8404		**Invalid value at parameter "Mode" of the Motion Control instruction**	
	16#0011	Invalid selection value	Correct the selection value; restart the command
	16#0015	Active/passive homing is not configured	Correct the configuration and download it to the controller; enable the axis and restart the command
	16#0017	Axis reversal is activated at the HW limit switch, despite the fact that the hardware limit switches are disabled	• Activate the hardware limit switch using the tag <Axis>.Config.PositionLimits_HW.Active = TRUE, restart the command • Correct the configuration and download it to the controller; enable the axis and restart the command
16#8405		**Invalid value at parameter "StopMode" of the Motion Control instruction**	
	16#0011	Invalid selection value	Correct the selection value; enable the axis again
16#8406		**Simultaneous forward and backward jogging is not allowed**	
	16#0001	-	Take steps to ensure that parameters "JogForward" and "JogBackward" do not have signal status TRUE simultaneously; restart the command.
16#8407		**Switching the axis with Motion Control instruction "MC_Power" is only permitted after disabling the axis.**	
	16#0001	-	Disable the active axis; it is then possible to switch to the other axis and enable it.

Configuration error

ErrorID	ErrorInfo	Description	Remedy
16#8600		**Parameterization of pulse generator (PTO) is invalid**	
	16#000B	Address is invalid	Correct the configuration of the PTO (Pulse Train Output) and download it to the controller
16#8601		**Parameterization of the high-speed counter (HSC) is invalid**	
	16#000B	Address is invalid	Correct the configuration of the HSC (High Speed Counter) and download it to the controller
16#8602		**Invalid parameter assignment of "Enable output"**	
	16#000D	Address is invalid	Correct the configuration and download it to the controller
16#8603		**Invalid parameter assignment of "Ready input"**	
	16#000D	Address is invalid	Correct the configuration and download it to the controller
16#8604		**Invalid "Pulses per motor revolution" value**	
	16#000A	The value is less than or equal to zero	Correct the configuration and download it to the controller
16#8605		**Invalid "Load distance per motor revolution" value**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#000A	The value is less than or equal to zero	

ErrorID	ErrorInfo	Description	Remedy
16#8606		**Invalid "Start / stop velocity" value**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
	16#0007	The start/stop velocity is greater than the maximum velocity	
16#8607		**Value for "Maximum velocity" is invalid**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
16#8608		**Invalid "Acceleration" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
16#8609		**Invalid "Deceleration" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
16#860A		**Invalid "Emergency stop deceleration" value**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
16#860B		**Value for position of the low SW limit switch is invalid**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
	16#0007	The position value of the low SW limit switch is greater than that of the high SW limit switch	
16#860C		**Value for position of the high SW limit switch is invalid**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with
	16#0005	The value is outside the number range (greater than $1e^{12}$)	

ErrorID	ErrorInfo	Description	Remedy
	16#0006	The value is outside the number range (less than -1e^{12})	instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#860D		**Invalid address of the low HW limit switch**	
	16#000C	Address of falling edge is invalid	Correct the configuration and download it to the controller
	16#000D	Address of rising edge is invalid	
16#860E		**Invalid address of the high HW limit switch**	
	16#000C	Address of falling edge is invalid	Correct the configuration and download it to the controller
	16#000D	Address of rising edge is invalid	
16#860F		**Invalid "home position offset" value**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0005	The value is outside the number range (greater than 1e^{12})	
	16#0006	The value is outside the number range (less than -1e^{12})	
16#8610		**Invalid "approach velocity" value**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0008	Velocity is greater than the maximum velocity	
	16#0009	Velocity is less than the start/stop velocity	
16#8611		**Invalid "Homing velocity" value**	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0008	Velocity is greater than the maximum velocity	
	16#0009	Velocity is less than the start/stop velocity	
16#8612		**Invalid address of the homing switch**	
	16#000C	Address of falling edge is invalid	Correct the configuration and download it to the controller
	16#000D	Address of rising edge is invalid	
16#8613		**During active homing, direction reversal at the hardware limit switch is activated although the hardware limit switches are not configured**	
	16#0001	-	Correct the configuration and download it to the controller

Internal errors

ErrorID	ErrorInfo	Description	Remedy
16#8FFF		Internal error	
	16#F0**	-	POWER OFF and POWER ON the CPU If this does not work, contact Customer Support. Have the following information ready: • ErrorID • ErrorInfo • Diagnostic buffer entries

See also

Using multiple axes with the same PTO (Page 3015)

Using online and diagnostics functions 12

12.1 General information about online mode

Online mode

In online mode, there is an online connection between your programming device / PC and one or more devices.

An online connection between the programming device/PC and the device is required, for example, for the following tasks:

- Testing user programs
- Displaying and changing the operating mode of the CPU
- Displaying and setting the date and time of day of the CPU
- Displaying module information
- Comparing blocks
- Hardware diagnostics

Before you can establish an online connection, the programming device/PC and the device must be physically or remotely connected. As an alternative, some devices support a simulation mode. In this case, a connection to the device is simulated via the PLCSIM virtual interface.

After establishing a connection, you can use the Online and Diagnostics view or the "Online tools" task card to access the data on the device. The current online status of a device is indicated by an icon to the right of the device in the project tree. You will find the meaning of the individual status icons in the relevant tooltip.

Note

Some online functions depend on the range of the installed software or whether a project is open.

Standby or hibernation of the programming device / PC

If the programming device / PC is changed to the standby or hibernation mode when there is an online connection, all online connections are terminated. When the programming device / PC wakes up from hibernation, the online connections are not automatically re-established.

Note that suddenly terminating an online connection can lead to loss of data or a connected device may interrupt program execution.

Performing an LED flash test

In many online dialogs you can perform an LED flash test, if the device connected online supports this feature. When you click on the "LED flashing" button, an LED flashes at the currently selected station. This feature is useful, for example, when you are not sure which device in the hardware configuration corresponds to the station currently selected in the software.

Read any additional information and learn about the possible limitations to the LED flash test in the respective device documentation.

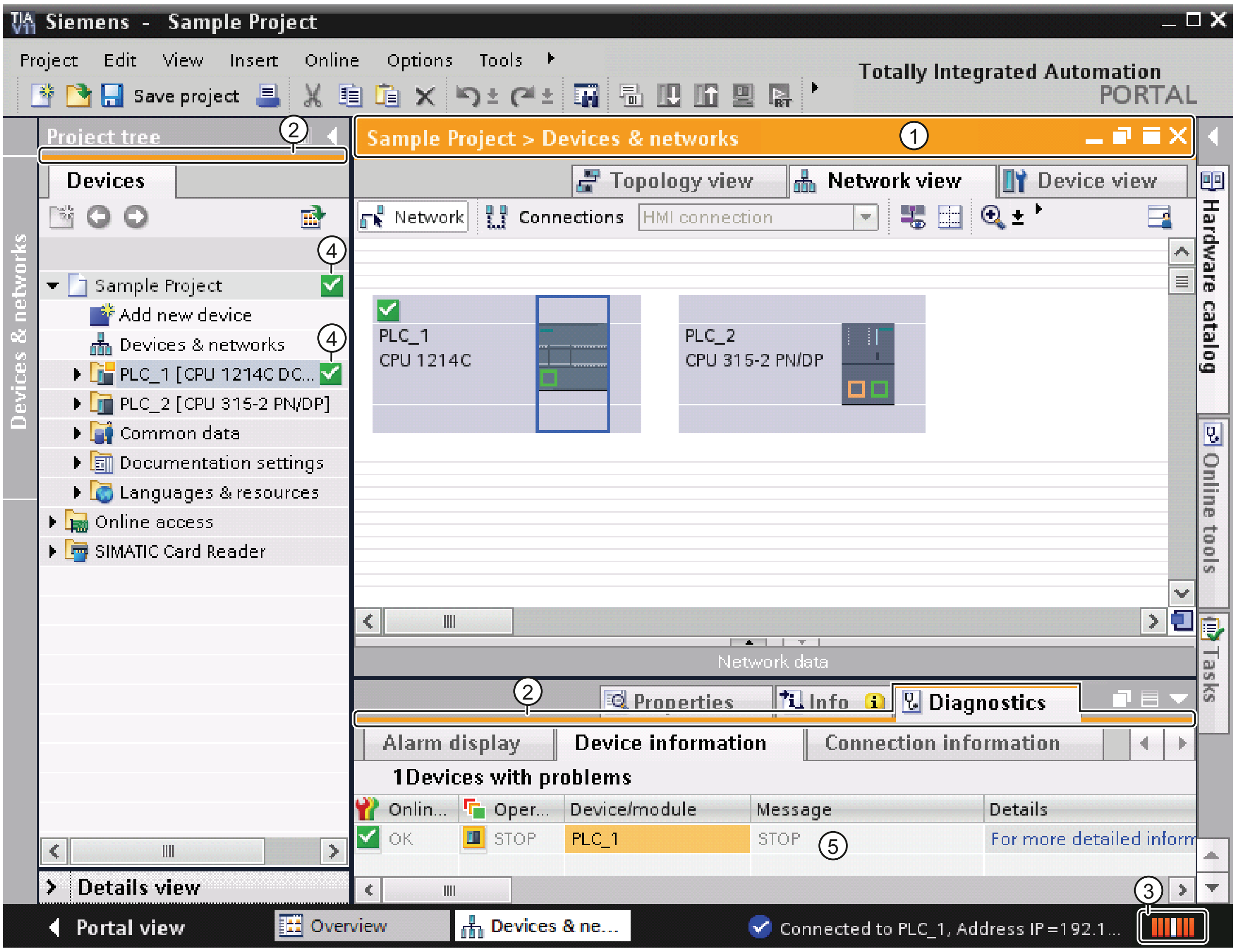
See also

View in online mode (Page 3067)

12.2 View in online mode

Online displays

After the online connection has been established successfully, the user interface changes. The following figure shows a device connected online and the corresponding user interface:



1Devices with problems

Onlin...	Oper...	Device/module	Message	Details
✓ OK	STOP	PLC_1	STOP	For more detailed inform

Connected to PLC_1, Address IP=192.1...

① The title bar of the active window now has an orange background.

② The title bars of inactive windows for the relevant station now have an orange line below them.

③ An orange, pulsing bar appears at the right-hand edge of the status bar. If the connection has been established but is functioning incorrectly, an icon for an interrupted connection is displayed instead of the bar. You will find more information on the error in "Diagnostics" in the Inspector window.

④ Operating mode symbols or diagnostics symbols for the stations connected online and their underlying objects are shown in the project tree. A comparison of the online and offline status is also made automatically. Differences between online and offline objects are also displayed in the form of symbols.

⑤ The "Diagnostics > Device information" area is brought to the foreground in the Inspector window.

Online connection abort

The online mode and its display are retained as long as at least one device is connected online. If the online connection to one or more devices aborts, the TIA Portal remains in online mode. The display of the TIA Portal changes to offline mode only when there is no longer an online connection to any device.

See also

General information about online mode (Page 3065)

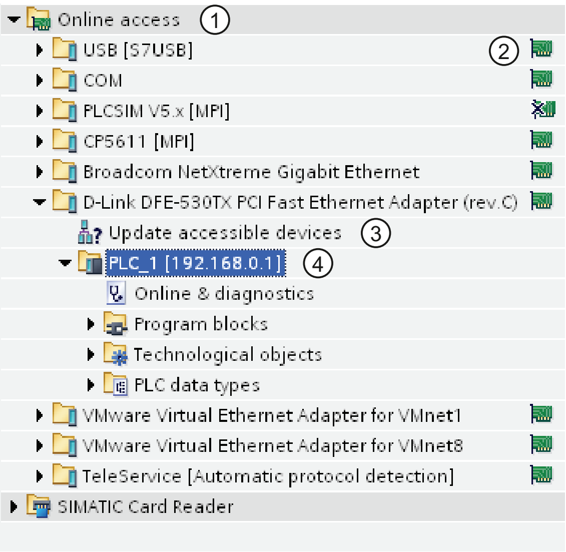
Basics of project data comparison (Page 233)

12.3 Online access

Online access of the project

In the "Online access" folder of the project tree, you will find all active interfaces of your programming device/PC. Each interface icon provides you with information on the status of the interface. You can also display the accessible devices and display and edit the properties of an interface using the shortcut menu.

The following figure shows the "Online access" folder in the project tree.



```
▼ Online access ①
    ▶ USB [S7USB]                                              ②
    ▶ COM
    ▶ PLCSIM V5.x [MPI]
    ▶ CP5611 [MPI]
    ▶ Broadcom NetXtreme Gigabit Ethernet
    ▼ D-Link DFE-530TX PCI Fast Ethernet Adapter (rev.C)
          Update accessible devices ③
        ▼ PLC_1 [192.168.0.1] ④
              Online & diagnostics
            ▶ Program blocks
            ▶ Technological objects
            ▶ PLC data types
    ▶ VMware Virtual Ethernet Adapter for VMnet1
    ▶ VMware Virtual Ethernet Adapter for VMnet8
    ▶ TeleService [Automatic protocol detection]
▶ SIMATIC Card Reader
```

① "Online access" folder in the project tree

All interfaces installed in the programming device/PC are displayed in the "Online access" folder.

② Status display for the interfaces

The current status of an interface is indicated by an icon to the right of the name. You can see the meaning of the icon in the tooltip.

③ Updating the list of accessible devices.

This function is available for each hardware interface of the programming device/PC. Software interfaces, such as a remote connection, do not offer this function.

④ Devices connected via the respective interface with the programming device/PC

The type of the respective device and its status are displayed by the preceding icon.

Displaying or updating accessible devices

You have the following options if you want to display all devices accessible online on your programming device/PC:

- Display of the accessible devices on a single interface of the programming device / PC in the project tree. In the project tree, you can also display additional information about the individual accessible devices.
- Display of the accessible devices of all interfaces in a list.

See also: Displaying accessible devices

Overview of icons for accessible devices

The accessible devices are identified with an icon according to their type and status. The following is an overview of all icons and their meaning.

	Icon for unidentified modules This icon is displayed whenever the identification of a module is not yet complete or when the identification of a module was not successful, for example, because the required online data could not be read.
	Icon for the following device types: • SIMOCODE pro devices • IE/PB links • CPs of PC systems • SCALANCE head modules • S7-300 and S7-400 CPs • PROFINET IO devices and PROFINET CPs • SCALANCE modules and gateways that could not be identified
	PROFINET IO devices, encoders, switchgear, sensors and identification systems that were replaced by similar devices because these could not be identified
	Icon for the following device types: • HMI devices • PROFINET IO devices of the HMI type if these could not be identified and were therefore replaced by a similar device
	PROFINET IO devices of the drive type that could not be identified and were therefore replaced by a similar device
	PROFINET IO devices of the development kit and network components type that could not be identified and were therefore replaced by a similar device
	PROFINET IO devices of the Teleservice adapter type that could not be identified and were therefore replaced by a similar device

See also

Opening the properties of an interface (Page 3073)

12.4 Displaying accessible devices

Accessible devices

Accessible devices are all devices connected to an interface of the programming device / PC and that are turned on. Devices that allow only restricted configuration using the currently installed products or that cannot be configured at all can also be displayed.

Displaying accessible devices on an interface of the programming device / PC in the project tree

To display accessible devices on a single interface of the programming device / PC, follow these steps:

1. Open the "Online access" folder in the project tree.
2. Click on the arrow to the left of the interface to show all the objects arranged below the interface.
3. Double-click on the "Update accessible devices" command below the interface.

All devices that are accessible over this interface are displayed in the project tree.

Displaying accessible devices in a list

To display the accessible devices on all available interfaces in an overview list, follow these steps:

1. Select the "Accessible devices" command in the "Online" menu.

   The "Accessible devices" dialog is displayed.
2. Select the type of interface from the "Type of the PG/PC interface" drop-down list. The "PG/PC interface" drop-down list then shows only the interfaces of the programming device / PC that match the selected interface type.
3. Select the required interface of the programming device / PC from the "PG/PC interface" drop-down list, for example an Industrial Ethernet adapter.

   If no devices are available on an interface, an unbroken connecting line is displayed between the programming device / PC and the device. If devices are accessible, an unbroken connecting line is shown and the devices accessible on the selected interface of the programming device / PC are displayed in a list.
4. If you have connected a new device in the meantime, click the "Refresh" button to refresh the list of accessible devices.
5. To go to a device in the project tree, select the device from the list of accessible devices and click the "Show" button.

   The interface to which the selected device is connected is shown as selected in the project tree.

Displaying additional information about the accessible devices in the project tree

To display additional information on the accessible devices in the project tree, follow these steps:

1. Click on the arrow to the left of one of the accessible devices in the project tree.

   All data available online, for example blocks and system data, is displayed for known devices.

12.5 Opening the properties of an interface

Introduction

For each interface, you can display and, in some cases, modify properties, for example the network type, address, and status.

Procedure

To open the properties, follow these steps:

1. Right-click on the required interface below "Online access" in the project tree.
2. Select the "Properties" command from the shortcut menu.

   A dialog containing the properties of the interface opens. On the left of the dialog, you will see the area navigation. You can view the current parameter settings in the individual entries in the area navigation and, if necessary, change them.

12.6 Establishing and canceling an online connection

Requirement

At least one PG/PC interface is installed and is physically connected to a device, for example with an Ethernet cable. As an alternative, it is also possible to establish a virtual connection using PLCSIM.

Go online

To establish an online connection, follow these steps:

1. In the project tree, select one or mote devices to which you want an online connection to be established.
2. Select the "Go online" command in the "Online" menu.

   If the device was already connected to a specific PG/PC interface, the online connection is automatically established to the previous PG/PC interface. In this case, you can ignore the following steps. If there was no previous connection, the "Go online" dialog opens.
3. Select the type of interface from the "Type of the PG/PC interface" drop-down list. The "PG/PC interface" drop-down list then shows only the interfaces of the programming device / PC that match the selected interface type.
4. Select the required interface of the programming device / PC from the "PG/PC interface" drop-down list, for example an Industrial Ethernet adapter.
5. In the "Connection to subnet" drop-down list, select the subnet via which the device is connected to the PG/PC interface. If the device is connected directly to the PG/PC interface, select the "(local) TCP/IP" setting. In this case, a direct connection is established to the device, without a network node, for example, an interposed switch.

   If you selected an MPI or PROFIBUS subnet, the bus parameters configured in the programming device/PC interface are applied at this point.
6. If the device is accessible via a gateway, select the gateway that connects the two subnets involved in the "1st gateway" drop-down list.

   If no devices are available on the interface, a broken connecting line is displayed between the programming device / PC and the device. If devices are accessible, an unbroken connecting line is shown and the devices accessible on the selected interface of the programming device / PC are displayed in a list.
7. Optional: Click the "Update" button to redisplay the list of accessible devices.
8. Optional: Click the "Flash LED" button on the left of the graphic to run an LED flash test. With this function, you can check that you have selected the correct device. The LED flash test is not supported by all devices.
9. Select your device in the "Accessible devices in the target subnet" table and confirm your selection with "Go online".

   The online connection to the selected target device is established.

Result

After the online connection has been established, the title bars of the editors change to orange. An orange activity bar is also shown in the title bar of an editor and in the status bar. In the project tree, status symbols show the difference between online and offline objects.

The connection path is stored for future connection attempts. It is no longer necessary to open the "Go online" dialog unless you want to select a new connection path.

Note

If no accessible device is displayed, select a different network access for the PG/PC interface or check the settings of the interface.

Canceling an online connection

To disconnect the existing online connection, follow these steps:

1. Select the device you want to disconnect from in the project tree.
2. Select the "Go offline" command in the "Online" menu.

See also

Connecting online with several devices (Page 3076)

View in online mode (Page 3067)

Assigning a temporary IP address (Page 3083)

Influence of user rights (Page 197)

12.7 Connecting online with several devices

You can establish an online connection to several devices at the same time without needing to select individual devices previously in the network view.

Requirement

- No device must be selected
- At least one PG/PC interface is installed and is physically connected to a device, for example with an Ethernet cable. As an alternative, it is also possible to establish a virtual online connection using PLCSIM or a remote connection.

Procedure

To establish an online connection to several devices at the same time, follow these steps:

1. Select the "Go online" command in the "Online" menu.

   The "Select devices" dialog opens with a table of all available devices.
2. Select the devices to which you want to establish an online connection in the "Go online" column.
3. Click the "Go online" button.

Result

Without any further prompt for confirmation, a connection is established to all selected devices if a connection was already established to the selected devices at least once. If there was no previous online connection, the "Go online" dialog opens. In this case, first configure the online connection as described in the section "Go online and disconnect online connection (Page 3074)".

See also

Establishing and canceling an online connection (Page 3074)

Assigning a temporary IP address (Page 3083)

12.8 Basics of assigning parameters for the PG/PC interface

Options for connecting to target systems

If the devices of the project are connected via different subnets, you assign a suitable network access to each PG/PC interface to be able to establish online connections to the target systems. The following interfaces are automatically supported:

- MPI
- PROFIBUS
- Industrial Ethernet (ISO and TCP/IP)

You can make various settings for the interfaces. The following sections explain the parameter settings you can make.

Note

Note that changes to interface parameters have a direct influence on the operating system and the programming device / PC. Remember that some parameter settings can only be changed if you have adequate user rights.

See also

Setting parameters for the Industrial Ethernet interface (Page 3079)

Setting parameters for the MPI and PROFIBUS interfaces (Page 3085)

12.9 Adding interfaces

You have the option of installing additional interfaces after installation of the TIA Portal.

Procedure

To install an interface at a later time and add it to the TIA Portal, follow these steps:

1. Install or update the drivers in the operating system once you have installed the interface hardware.
2. Close the TIA Portal if it is still open.
3. Open the Windows control panel.
4. Open the entry "Setting the PG/PC Interface" in the Control Panel.

   The "Setting the PG/PC Interface" dialog opens.
5. Make any necessary changes to the interface configuration and confirm them with "OK". You have to click "OK", even if you have not made any changes.
6. Restart the TIA Portal.

Result

The newly installed interface is now displayed in the project tree under the "Online access" folder.

12.10 Setting parameters for the Ethernet interface

12.10.1 Setting parameters for the Industrial Ethernet interface

Options in the parameter settings for the Industrial Ethernet interface

When setting parameters for the Industrial Ethernet interface, you have the following options:

- Parameters dependent on the operating system

  The Industrial Ethernet interface has parameters that are set in the operating system and are valid for all connected devices. These parameter settings are only displayed here, they can, however, be changed in the network settings of the operating system.
- Parameters that can be set in the software

Note

Note that changes to interface parameters have a direct influence on the operating system and the programming device / PC. Remember that some parameter settings can only be changed if you have adequate user rights.

Parameters for the Industrial Ethernet interface

The following table contains an overview of the parameters of the Industrial Ethernet interface that are set by the operating system and can be changed by the user.

Parameter settings that cannot be changed	Parameters that can be set
MAC address	Fast acknowledge at the IE-PG access and for TCP/IP
DHCP server activated/deactivated	Timeout at the IE-PG access and for TCP/IP
APIPA activated/deactivated	LLDP
IP address	Additional, dynamic IP addresses for the network adapter
Subnet mask	-
DNS addresses	-
DHCP addresses	-

See also

Basics of assigning parameters for the PG/PC interface (Page 3077)

Displaying operating system parameters (Page 3080)

Connecting the PG/PC interface to a subnet (Page 3081)

Setting parameters for the Ethernet interface (Page 3081)

Assigning a temporary IP address (Page 3083)

Managing temporary IP addresses (Page 3083)

Influence of user rights (Page 197)

12.10.2 Displaying operating system parameters

The Ethernet interface is part of the operating system. All parameters of the network adapter can therefore be adapted in the network settings of the operating system.

You can display the following parameters in the software:

- Physical address of the network adapter
- Assignment of the IP address by a DHCP server activated or deactivated
- Assignment of a private IP address by the operating system activated or deactivated
- Current static IP address
- Assigned subnet mask
- DNS addresses
- DHCP addresses

If you want to modify the parameter settings, please refer to the documentation of the operating system or the network adapter.

Displaying current parameters of the Ethernet interface

To display the current parameters of the Ethernet interface, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.

   The dialog for configuring the interface opens.
3. Select "Configurations > Industrial Ethernet" in the area navigation.

See also

Setting parameters for the Ethernet interface (Page 3081)

12.10.3 Connecting the PG/PC interface to a subnet

If you have created several subnets, you can specify the subnet to which the Ethernet interface is connected.

Procedure

To select the subnet to which the Ethernet interface is connected, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.

   The dialog for configuring the interface opens.
3. Go to "General > Assignment" and select the subnet to which you want to connect the Ethernet interface of the programming device / PC in the "Connection to subnet" drop-down list.
4. Close the dialog with "OK".

12.10.4 Setting parameters for the Ethernet interface

You can adapt some parameter settings relating to the network protocol directly in the software.

Requirement

You must have adequate user rights.

See also: Influence of user rights (Page 197).

Procedure

To change parameter settings relating to the network protocol, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.

   The dialog for configuring the interface opens.
3. Select "Configurations > IE-PG access" to adapt the protocol settings relevant to network management.
   - Select the "Fast acknowledge" check box to achieve faster reaction times with smaller network packets.
   - From the "Timeout" drop-down list, select the maximum time that can elapse before a network node is detected.
4. To activate the LLDP protocol and discover the network topology more accurately, set the "LLDP active" check box in "Configurations > LLDP".
5. Select "Configurations > TCP/IP" to adapt the TCP/IP protocol for network traffic during runtime.
   - Select the "Fast acknowledge" check box to achieve faster reaction times with smaller network packets.
   - From the "Timeout" drop-down list, select the maximum time that can elapse before there is a timeout during communication with a network node.

See also

Influence of user rights (Page 197)

Displaying operating system parameters (Page 3080)

12.10.5 Assigning a temporary IP address

Adding a dynamic IP address

If the IP address of a device is located in a different subnet from the IP address of the network adapter, you will first need to assign an additional IP address with the same subnet address as the device. Only then is communication between the device and the programming device / PC possible.

The assignment of an additional temporary IP address is also proposed automatically if you want to perform an online action and the current IP address of the programming device/PC is not yet in the correct subnet.

A temporarily assigned IP address remains valid until the next time the programming device/PC is restarted or until you delete it manually.

Note

You require adequate permissions to be able to assign a temporary IP address.

See also: Influence of user rights (Page 197)

See also

Managing temporary IP addresses (Page 3083)

12.10.6 Managing temporary IP addresses

If the IP address of a device is located in a different subnet from the current static IP address of the network adapter, the network adapter temporarily assigns a suitable IP address from the subnet of the device.

You can display all temporarily assigned addresses and delete them. Note that IP addresses that you manually assigned in the operating system are not displayed in the TIA Portal.

Requirement

To delete, you require adequate permissions.

Procedure

To display and delete temporarily assigned addresses, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.

   The dialog for configuring the interface opens.
3. Select "Configurations > IE-PG access".

   A table with the assigned IP addresses is displayed.
4. Click the "Delete project-specific IP addresses" button to delete all the IP addresses at one time.

See also

Influence of user rights (Page 197)

12.10.7 Resetting the TCP/IP configuration

If you have changed the TCP/IP protocol settings, you can reset them to the defaults.

Procedure

To restore the TCP/IP configuration to the default settings, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.

   The dialog for configuring the interface opens.
3. Select "Configurations > TCP/IP".
4. Click the "Standard" button to reset all the settings.

12.11 Setting parameters for the MPI and PROFIBUS interfaces

12.11.1 Setting parameters for the MPI and PROFIBUS interfaces

Possible parameter settings for the MPI and PROFIBUS interfaces

The following parameter settings can be made for the MPI and PROFIBUS interfaces:

- Automatic configuration: You can use automatic detection functions to find out whether a device is connected to the PG/PC interface over PROFIBUS or MPI.
- Selecting a default configuration for PROFIBUS or MPI that can be adapted later.

Device- and network-related settings for MPI and PROFIBUS

You can set device- and network-related parameters for MPI and PROFIBUS interfaces. Device-related parameters are local settings for the interface. Network-related parameters, on the other hand, must match up on all devices.

MPI interface parameters you can modify

You can adapt the following default parameters for the MPI interface:

Device-related parameters	Network-related parameters
Is the only master	Highest address
Own address	Transmission rate
Timeout	

PROFIBUS interface parameters you can modify

You can adapt the following default parameters for the PROFIBUS interface:

Device-related parameters	Network-related parameters
Is the only master	Highest address
Own address	Transmission rate
Timeout	Profile
	Bus parameters
	Number of masters on bus
	Number of slaves on bus

See also

Basics of assigning parameters for the PG/PC interface (Page 3077)

12.11.2 Setting MPI or PROFIBUS interface parameters automatically

Setting up automatic bus parameter detection

If you select an interface with automatic detection of the bus parameters (for example CP 5611 (Auto)), you can connect the programming device or PC to MPI or PROFIBUS without needing to set bus parameters. At a transmission speed lower than 187.5 Kbps, you may, however, have waiting times of up to one minute.

Requirement

- Masters that distribute bus parameters cyclically are connected to the bus.
- In PROFIBUS networks, the cyclic distribution of the bus parameters must be enabled.

Procedure

To enable automatic bus parameter detection, follow these steps:

1. Select the interface in the project tree.
2. Select the "Properties" command in the shortcut menu of the interface.

   The dialog for configuring the interface opens.
3. Go to "General > Configurations > Active configuration" and select the setting "Automatic protocol detection".
4. Go to "Configurations > Auto configuration > Local settings" and select the address of the PG/PC interface in the "Own address" drop-down list.
5. If you then want to display the current bus settings, click the "Network detection" button.

See also

Setting parameters for the MPI interface (Page 3087)

Setting parameters for the PROFIBUS interface (Page 3089)

12.11.3 Setting parameters for the MPI interface

Changing the parameter settings of the MPI interface

The network-related parameters and bus parameters for the MPI network can be adapted. You should first select a default setting and then adapt this to the specific situation.

Setting defaults for the MPI interface

To adapt the parameters of the MPI interface, follow these steps:

1. Select the interface in the project tree.
2. Select the "Properties" command in the shortcut menu of the interface.

   The dialog for configuring the interface opens.
3. Go to "General > Assignment" and select the subnet with which you want to connect the interface in the "Connection to subnet" drop-down list.
4. Under "General > Configuration", select a default for the device and network-related parameters. The defaults are suitable for most configurations. Select one of the following settings:
   - Automatic protocol detection

     You can connect the programming device to MPI or PROFIBUS without having to set bus parameters. At a transmission speed lower than 187.5 Kbps, you may, however, have waiting times of up to one minute. Prerequisite for the automatic detection is a connection to the bus master, which distributes the bus parameters cyclically. With PROFIBUS subnets, cyclic distribution of bus parameters may not be deactivated (default PROFIBUS network setting).
   - MPI

     The "MPI" transmission protocol is selected. Typical parameters are set that are adequate for most configurations. You can change the parameters to your needs, however.
   - PROFIBUS

     The "PROFIBUS" transmission protocol is selected. Typical parameters are set that are adequate for most configurations. You can change the parameters to your needs, however.

Changing the default parameter settings

To adapt the default settings to your requirements, change the parameter setting where necessary in "Configurations > MPI".

You can set the following device-related parameters:

- Is only master

  An additional verification function to prevent bus disruptions when connecting the PG/PC to the network is disabled because the programming device or PC is the only master on the bus.

  – Do not enable this option unless you have only connected slaves to your programming device or PC.

  – If the "Is only master" check box is enabled, it is not possible to identify the directly connected device in the "Accessible devices" window.

- Own address

  This setting relates to the programming device or PC on which you call up the parameter settings of the interface. Set the local device address of your programming device or PC here.

  – This address must be unique throughout the network.

  – The programming device or PC is addressed using this address in the MPI network.

- Check

  This enables an additional safety function to prevent bus disruptions when connecting the PG/PC to the network. The driver checks whether the local address is already being used by another station. Active as well as passive stations are taken into consideration in this case. The driver monitors this on the PROFIBUS. The connection of the PG/PC to the network will take longer with the automatic check. To use the check, the driver must support the function. Furthermore, the "Is only master" option must not be selected.

- Timeout

  Set a higher timeout value if, for example, you have problems with long response times on the network.

You can set the following network-related parameters:

- Highest address:

  Select the configured highest device address. Make sure that the same highest device address is set for all devices of a PROFIBUS or MPI network.

- Transfer rate:

  Here, you select the transmission speed to be used on the MPI network.

See also

Setting MPI or PROFIBUS interface parameters automatically (Page 3086)

12.11.4 Setting parameters for the PROFIBUS interface

Changing the parameter settings of the PROFIBUS interface

The network-related parameters and bus parameters for the PROFIBUS network can be adapted more precisely. You should first select a default setting and then adapt this to the specific situation.

Setting defaults for the PROFIBUS interface

To adapt the parameters of the PROFIBUS interface, follow these steps:

1. Select the interface in the project tree.
2. Select the "Properties" command in the shortcut menu of the interface.

   The dialog for configuring the interface opens.
3. Go to "General > Assignment" and select the subnet with which you want to connect the interface in the "Connection to subnet" drop-down list.
4. Under "General > Configuration", select a default for the device and network-related parameters. The defaults are suitable for most configurations. Select one of the following settings:
   - Automatic protocol detection

     You can connect the programming device to MPI or PROFIBUS without having to set bus parameters. At a transmission speed lower than 187.5 Kbps, you may, however, have waiting times of up to one minute. Prerequisite for the automatic detection is a connection to the bus master, which distributes the bus parameters cyclically. With PROFIBUS subnets, cyclic distribution of bus parameters may not be deactivated (default PROFIBUS network setting).
   - MPI

     The "MPI" transmission protocol is selected. Typical parameters are set that are adequate for most configurations. You can change the parameters to your needs, however.
   - PROFIBUS

     The "PROFIBUS" transmission protocol is selected. Typical parameters are set that are adequate for most configurations. You can change the parameters to your needs, however.

Changing the default parameter settings

To adapt the default settings to your requirements, change the parameter setting where necessary in "Configurations > PROFIBUS".

You can set the following device-related parameters:

- Is only master

  An additional verification function to prevent bus disruptions when connecting the PG/PC to the network is disabled because the programming device or PC is the only master on the bus.

  – Do not enable this option unless you have only connected slaves to your programming device or PC.

  – If the "Is only master" check box is enabled, it is not possible to identify the directly connected device in the "Accessible devices" window.

- Own address

  This setting relates to the programming device or PC on which you call up the parameter settings of the interface. Set the local device address of your programming device or PC here.

  – This address must be unique throughout the network.

  – The programming device or PC is addressed using this address in the PROFIBUS network.

- Check

  This enables an additional safety function to prevent bus disruptions when connecting the PG/PC to the network. The driver checks whether the local address is already being used by another station. Active as well as passive stations are taken into consideration in this case. The driver monitors this on the PROFIBUS. The connection of the PG/PC to the network will take longer with the automatic check. To use the check, the driver must support the function. Furthermore, the "Is only master" option must not be selected.

- Timeout

  Set a higher timeout value if, for example, you have problems with long response times on the network.

You can set the following network-related parameters:

- Highest address:

  Select the configured highest device address. Make sure that the same highest station address is set for all devices of a PROFIBUS network.

- Transfer rate:

  Here, you select the transmission speed to be used on the PROFIBUS network.

- Profile:

  You have a choice of four alternatives for the PROFIBUS settings. "DP", "Standard" and "Universal (DP/FMS)" are predefined settings that you cannot change. If you select "User-defined", you can adapt the bus parameters yourself.

- If you have selected "User-defined", go to "Configurations > PROFIBUS > Bus parameters" in area navigation.
- If you have selected one of the defaults (DP, Standard or Universal (DP/FMS)), you should select the "Include" check box in "Configurations > PROFIBUS > Bus parameters > Additional parameters". You can then set the number of masters and slaves on the bus. This allows a more precise calculation of the bus parameters and potential bus disruptions can be prevented. The option cannot be selected with a user-defined profile.

See also

Overview of the bus parameters for PROFIBUS (Page 3091)

Setting MPI or PROFIBUS interface parameters automatically (Page 3086)

12.11.5 Overview of the bus parameters for PROFIBUS

Introduction

The PROFIBUS subnet will only function problem-free if the parameters for the bus profile are matched to one another. You should therefore only change the default values if you are familiar with how to configure the bus profile for PROFIBUS.

It may be possible for the bus parameters to be adjusted depending on the bus profile. The offline values of the bus parameters are always shown even if you are online and linked to the target system.

The displayed parameters are valid for the entire PROFIBUS subnet.

Meaning of the individual parameters

- Tslot: Wait-to-receive time (slot time)

  The wait-to-receive time (slot time) defines the maximum time the sender will wait to receive a response from the addressed partner.
- Max. Tsdr: Maximum protocol processing time (max. station delay responder)

  The maximum protocol processing time defines the time after which the responding device must have processed the protocol.
- Min. Tsdr: Minimum protocol processing time (min. station delay responder)

  The minimum protocol processing time specifies the minimum time required by the responding device to process the protocol.
- Tset: Trigger time (setup time)

  The trigger time is the time that may lapse between the reception of a data frame frame and the reaction to it.

- Tqui: Quiet time for modulator

  The quiet time for modulator specifies the time required to change from sending to receiving.
- GAP factor: GAP update factor (GAP factor)

  The GAP factor specifies the number of token rotations before a new device is included in the token ring.
- Retry limit: Maximum number of repeated call attempts (retry limit)

  This parameter defines the maximum number of attempts made to reach a device.
- Trdy: Ready time

  The ready time is the time for an acknowledgment or response.
- Tid1: Idle time 1

  Idle time 1 specifies the delay time after receiving a response.
- Tid2: Idle time 2

  Idle time 2 specifies the delay time after sending a call without a response.
- Ttr: Target rotation time

  The target rotation time is the maximum time made available for a token rotation. During this time, all active devices (masters) receive the token once. The difference between the desired token round-trip time and the actual token round-trip time decides how much time is left for masters to send data frames to the slaves.

  As the minimum target rotation time (Ttr), select a value = 5000 times the HSA (Highest Station Address).
- Watchdog: Watchdog

  The watchdog time specifies the time after which a device must be addressed.

  As the minimum watchdog time, select a value = 6250 times the HSA.

Note

If you want to create a user-defined bus profile, please note that the minimum target rotation time (Ttr) should be 5000 times the HSA (highest PROFIBUS address). The minimum watchdog time should also be 6250 times the HSA.

See also

Setting parameters for the PROFIBUS interface (Page 3089)

12.11.6 Resetting the MPI or PROFIBUS configuration

If you have changed the MPI or PROFIBUS protocol settings, you can reset them to the defaults.

Procedure

To restore the MPI or PROFIBUS configuration to the default settings, follow these steps:

1. Select the MPI/PROFIBUS interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.

   The dialog for configuring the interface opens.
3. Select "Configurations > MPI" or "Configurations > PROFIBUS", depending on the interface properties you want to reset.
4. Click the "Standard" button to reset all the settings.

12.12 Establishing a remote connection with TeleService

12.12.1 Basics of working with TeleService

12.12.1.1 Introduction to TeleService

Introduction

TeleService gives your controller telecommunication capability. You can manage, control and monitor distributed plants centrally by means of remote connections.

Scope of functions

TeleService allows you to use the range of TIA portal functions via a telephone network by establishing a remote connection to a remote system. The online connection allows you to edit a remote system in the usual way with the TIA Portal.

Advantages

Using TeleService offers the following advantages:

- You can easily access even remote sections of plants and include them in a complete system.
- You can offer rapid help and support in the event of faults in a remote system without having to go there yourself.
- You can employ your resources effectively.
- It significantly reduces costs.
- It can significantly reduce plant downtimes.
- It improves the efficiency of your plant.

See also

TeleService functionality (Page 3095)

12.12.1.2 TeleService functionality

TeleService functionality

TeleService offers the following functionality:

- Access to remote systems (remote maintenance):
  You can manage, control, and monitor distributed plants centrally by means of remote connections.
  This is possible with a CPU S7-1200 or CPU S7-300/400 and a TS Adapter MPI or a TS Adapter IE.
- Establishing connections from and to remote systems (PG-AS remote link):
  You can use PRODAVE MPI V5.0 and higher to establish a remote connection to a remote system, and the communication instruction "PG_DIAL" to establish a remote connection from a remote system.
  This is possible with a CPU S7-300/400 or a TS Adapter MPI.
- Data exchange between plants (AS-AS remote link):
  The communication instruction "AS_DIAL" allows two automation systems to exchange process data via the telephone network.
  This is possible with a CPU S7-300/400 and a TS Adapter MPI.
- Sending an SMS from a system:
  An automation system can send a message (SMS) via a GSM radio modem using the communication instruction "SMS_SEND" .
  This is possible with a CPU S7-300/400 or a TS Adapter MPI.
- Sending an email from a plant
  With the communications instruction "AS_MAIL" an automation system can send an email.
  This is possible with a CPU S7-31x-2PN/DP or a CPU 41x-3PN/D and a TS Adapter IE.

See also

Basics of using a TS adapter (Page 3100)

Supported telephone networks and modems (Page 3096)

12.12.1.3 Telephone book at TeleService

Introduction

By double clicking on the "Telephone book" folder in the project navigation, the telephone book editor displaying the global telephone book is opened.

Global telephone book properties

The global telephone book in TeleService is used to manage the specific system data required for establishing a remote connection.

The first time you open the phone book, an empty phone book will be displayed with all available columns; in all other cases, the last phone book to be edited will be displayed.

You can enter any number of plants in a phone book. Systems contain the data required for establishing a remote connection, for example, the name and location of the device and the phone number to be dialed, along with all country specific details.

The respective TS adapters used are distinguished by color, depending on if a TS Adapter MPI or a TS Adapter IE is used for establishing the connection.

12.12.2 Telephone networks and modems

12.12.2.1 Supported telephone networks and modems

Telephone networks which can be used

TeleService can be used with digital networks (ISDN), analog networks and radio networks (with GSM technology). The table below shows the transmission duration scaled to the transmission duration of the MPI card (CP 5611 = 1x) and as a function of the TS Adapter MPI used:

Connection:	with TS Adapter I V5.0:	with TS Adapter I V5.1/V5.2:	with TS Adapter II V1.0:
Direct connection (COM, 19.2 Kbps)	8 x	8 x	-
Direct connection (USB)	-	-	2.5 x
ISDN network (64 Kbps)	16 x	11 x	8 x
Analog network (28.8 Kbps)	32 x	14 x	11 x
Radio network (9.6 Kbps)	150 x	40 x	30 x

Restrictions

Communication with the TS Adapter via the CAPI interface is not possible with internal ISDN cards or with PCMCIA cards unless you have a virtual COM interface from the modem manufacturer.

Modem support

TeleService has been implemented to be independent of the modem. This means that all standard modems (Hayes-compatible/AT commands) which can be installed in the Windows Control Panel can also be used by TeleService.

The basic requirement is for a physical/virtual COM interface. The choice of modem type is determined primarily by the existing hardware of the programming device/PC and the telephone network to be used.

The following modem types/media are supported:

- Modems (external modems at the COM interface, internal modems and PCMCIA cards)
- External ISDN adapter at the COM interface or USB interface
- Internal ISDN adapter with virtual COM interface (for example AVM CAPI port)
- External ISDN modems (ISDN adapter with integrated analog modem functionality) at the COM interface or USB interface
- Radio network modems with GSM technology, PCMCIA adapter card or data cable and mobile phone

Gateways

Gateways between the various telephone networks are in principle possible. Remote connections from an ISDN adapter to an analog modem and vice versa only function with special ISDN telephone adapters.

Performance in telephone networks

The data throughput of a remote connection depends on the modem and telephone network used and the quality of the telephone line.

This version currently supports one remote connection to a TS Adapter.

12.12.2.2 Installing the local modem

Introduction

If you have already installed a modem for data transfer in your operating system, this modem can also be used for TeleService.

If a modem has not yet been installed for your operating system, one must be installed before you can establish a remote connection with TeleService.

Procedure

Proceed as follows:

1. Make sure your programming device/PC and the modem are switched off.
2. Physically connect the external modem to a COM or USB interface on your programming device/PC. You can also install an internal modem or a PCMCIA card in accordance with the manufacturer's specifications.
3. Now switch on the modem and then the programming device or the PC.

Result

Plug-and-play modems are recognized and installed automatically by the operating system. Dialogs will take you through the installation procedure.

Note

Modems without plug-and-play

If your modem is not recognized automatically when switched on, you will have to install it yourself using the Control Panel.

Please refer to the information in the documentation supplied with your modem.

12.12.2.3 Connecting and configuring the remote modem

Introduction

A modem must also be connected to the remote system before you can work with TeleService. This modem is designated the "remote modem".

Configuring the remote modem

The modem receives all parameters required for operation from the TS Adapter connected. These include data for initializing the modem and settings for serial transmission between the TS Adapter and the modem.

The data required for the remote modem is specified during the configuration of the TS Adapter.

The modem may be internal or external depending on the TS Adapter used.

How to connect a TS Adapter with an internal modem

1. Switch off the TS Adapter.
2. Connect the TS Adapter to the automation system.
3. Connect the TS Adapter to the telephone line.
4. Switch on the TS Adapter.

How to connect a TS Adapter with an external modem

1. Switch off the modem.
2. Connect the TS Adapter to the automation system.
3. Connect the TS Adapter to the modem using a modem cable.
4. Connect the modem to the telephone line.
5. Switch on the modem.
6. Switch on the TS Adapter.

Note

Please note the following information on configuring the remote modem:

- The default parameters for the modem and the serial port set in the TS Adapter should in most cases ensure successful operation; changes in the parameter assignment will only be needed in rare cases.
- You need only change the parameter assignment of the TS Adapter if a modem connection is not established or if factory settings are to be adapted or optimized.
- TS Adapter parameter assignment can be changed via either a direct or a remote connection.

12.12.2.4 Initialization string requirements

Introduction

The initialization string is a string composed of AT commands (quasi-standard commands for modems) used to initialize the modem connected to the TS Adapter.

Initialization string requirements

The following properties must be specified in the string for initializing the modem:

- The modem outputs feedback messages.
- The feedback messages are output in clear text.
- The DCD signal is only activated when a connection exists.
- The speed of transmission between the TS Adapter and modem is not changed after the connection has been established.
- The RTS/CTS protocol has been activated as a flow control between the TS Adapter and the programming device or PC.
- Automatic call acceptance by the modem has been activated.

12.12.3 Using a TS adapter to establish a remote connection

12.12.3.1 Basics of using a TS adapter

Using a TS Adapter for TeleService

A TS Adapter is needed for establishing a remote connection using TeleService.

The TS Adapter is used to prepare an automation system for use with TeleService by connecting it to a telephone network via a modem. The TS adapter has an integrated parameter memory in which a parameter set for TeleService operation is stored.

With the function "export adapter parameter", different parameter sets can be saved in external files, and reloaded in the TS Adapter via the function "import adapter parameter".

The figure below shows how a remote connection can be established with variousTS Adapter:

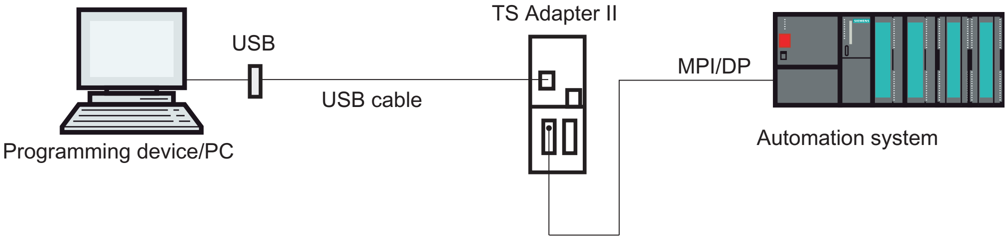


Figure 12-1 Establishing the remote connection

You can choose between a number of different TS Adapters, each of which offers a different functionality and different connection options.

Overview of possible TS Adapter:

The TS Adapter comes in the following versions:

- TS Adapter I (also designated "TS Adapter MPI")
- TS Adapter II (also designated "TS Adapter MPI")
- TS Adapter IE Standard (also designated "TS Adapter IE")
- TS Adapter IE Basic (also designated "TS Adapter IE")

Designation "TS Adapter"

In the following pages, the designation "TS Adapter" stands for all versions. The relevant product designation is listed beside information which only applies to a specific version, e.g. "TS Adapter I", "TS Adapter II", "TS Adapter IE Standard" or "TS Adapter IE Basic".

Note

For more detailed information on your TS Adapter, please refer to the documentation supplied with it.

See also

Short description of the TS adapter MPI (Page 3101)

Short description of the TS adapter IE (Page 3109)

Exporting adapter parameters (Page 3108)

Importing adapter parameters (Page 3108)

12.12.3.2 Installing TS adapter software

Requirement

A TS Adapter is needed for establishing a remote connection using TeleService.

Software for TS Adapter

The software required for running a TS Adapter is installed with the TIA portal.

No additional software needs to be installed to establish a remote connection with TeleService.

12.12.3.3 TS adapter MPI

Short description of the TS adapter MPI

TS Adapter MPI:

The designation "TS Adapter MPI" is a collective term for allTS Adapter with an MPI/DP interface.

The TS Adapter MPI comes in the following versions:

- As TS Adapter I (parameters cannot be assigned via the TIA portal)
- as TS Adapter II

The tables below provide a short description of the functionalities. For detailed information on your TS Adapter, please refer to the documentation supplied with it.

TS Adapter I	
Direct connection via the serial port. The firmware cannot be replaced.	
Version:	Main expansions:
V3.0	-
V5.0	Access protection
V5.1	Access protection, network type AUTO
V5.2	Access protection, network type AUTO, sending SMS messages

TS Adapter II:
Direct connection via the Universal Serial Bus (USB). Replaceable firmware. Modem integrated or external. The TS Adapter II switches automatically between the modems. As long as no external modem is connected, the adapter will use the internal modem.
There are two variants:
• With internal analog modem. An external modem can also be connected to the RS232 port.
• With internal ISDN adapter. An external modem can also be connected to the RS232 port.

Use of the designation "TS Adapter"

For TeleService the designation "TS Adapter" is the generalization for all versions. The relevant product designation is listed beside information which only applies to a specific version of a TS Adapter, e.g. "TS Adapter I", "TS Adapter II", "TS Adapter IE Standard" or "TS Adapter IE Basic".

How the TS adapter MPI works

How the TS Adapter MPI Works

In line with the configuration, the TS Adapter MPI connects the serial port or USB port of your programming device/personal computer (direct connection) or the serial port of a modem (modem connection) to the MPI/PROFIBUS network of your automation system.

The TS Adapter MPI has a non-volatile memory. Parameters for the following functions are stored in this memory:

- The MPI/PROFIBUS network (network parameters)
- The mode of the modem used
- The serial port to the modem
- Access protection

Default parameter assignment

The TS Adapter comes with default parameter assignment. The parameters can be set and saved to the non-volatile memory of the TS adapter in a parameter assignment session.

When "Direct connection" is configured, the TS Adapter will only use the network parameters for access to the MPI/PROFIBUS network.

In the "Modem connection" configuration, all the parameters stored on the TS Adapter will be activated.

Note

For more detailed information on the configuration of your TS Adapter, please refer to the documentation supplied with it.

Operating a TS adapter MPI in direct connection mode

Direct connection with TS Adapter MPI

The direct connection is used to assign the parameters of the TS adapter MPI. The same configuration also allows you to go online in the TIA portal and thereby check the assigned MPI/PROFIBUS parameters for bus compatibility. This means that (as with a PC adapter) SIMATIC S7/C7 systems can be accessed via the MPI/DP interface without an MPI/PROFIBUS module occupying a slot for a programming device/PC.

Access protection for the TS Adapter is not active in direct connection configuration. This means that the parameter assignment of the TS Adapter can be changed without any problems, for example by importing adapter parameters.

Note

Display the TS adapter MPI in the TIA portal

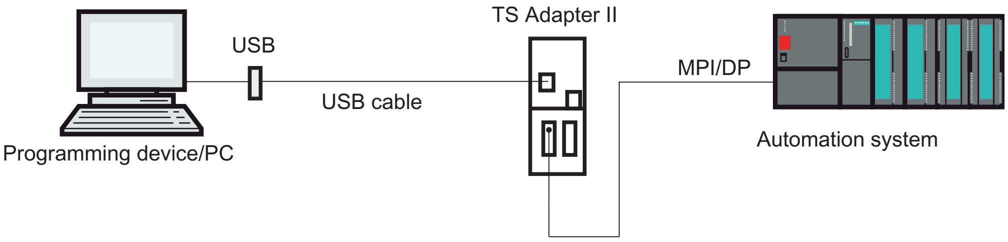
As soon as you have connected a TS adapter MPI with the PG/PC via the UBS interface, the folder "TS adapter" is displayed in the project navigation.

When you open the folder, you can assign the parameters of the connected TS adapter MPI via the following dialog.

Establishing the direct connection for TS Adapter MPI

Direct connection mode means there is a direct connection via the TS Adapter MPI between the programming device/personal computer on which TeleService is installed and the automation system. No modem is required.

The figure below shows the configuration of the TS Adapter MPI with a direct connection.



Operating a TS adapter MPI in modem connection mode

Introduction to the modem connection with TS Adapter MPI

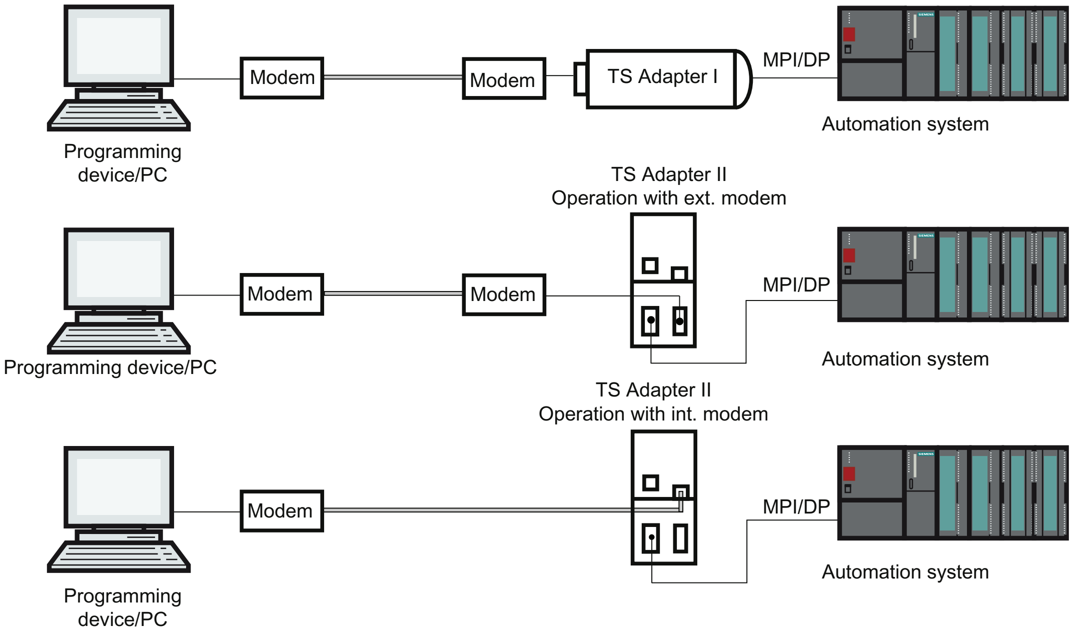
This configuration allows you to dial into a remote system. To do this, you establish a remote connection to a remote system using TeleService on a telephone network. You can then work with the selected system as usual, with the TIA portal, over the established modem connection.

Establishing a modem connection with TS Adapter MPI

This connection between the programming device or PC on which TeleService is installed and the automation system in which the TS Adapter MPI is inserted in the MPI/DP interface is made through a modem route.

The configuration therefore includes the programming device or PC via the telephone network and the TS Adapter MPI on the MPI/DP interface of the automation system.

The figure below shows the configuration of the TS Adapter MPI with a modem connection.



Note

Parallel operation between direct and modem connection

The TS adapter II has two connections for communication with PG/PC, both of which can be connected at the same time. At the same time, connect the USB interface with the PG/PC and the modem interface with the telephone network.

In this configuration you can either use the direct or the modem connection.

A parallel operation is not possible!

TS adapter MPI configuration options

Useful information on configuring the TS Adapter MPI

The TS Adapter MPI can be configured in both direct connection mode and via an existing remote connection.

The following parameter assignment options are available:

- Reconfiguration (Page 3106)
- Restoring default parameter assignment (Page 3107)
- Importing adapter parameters (Page 3108)
- Exporting adapter parameters (Page 3108)
- Setting up access protection (Page 3117)

Parameter assignment

Configure your TS Adapter in accordance with the documentation supplied with the TS Adapter. It will detail the exact procedure for parameter assignment.

Note

Please note the following when configuring the TS Adapter MPI

- If you change the current parameter settings when there is an established remote connection, there is a risk it will not subsequently be possible to establish a modem connection with the modified parameters. The TS Adapter MPI can in this case only be configured in direct connection mode.
- This means that either parameter assignment must be carried out with a programming device/personal computer at the plant location or the TS Adapter MPI must be brought to the location of the local programming device/personal computer in order to be configured.

Positive acknowledgement

During parameter assignment, the data is written to the non-volatile memory of the TS Adapter MPI. The parameter assignment process is not acknowledged positively until all precautions have been taken to ensure that parameter changes have been carried out correctly and will thus survive a power failure.

Changes become effective for the TS Adapter MPI as follows:

- The serial parameters, the modem parameters and the parameters for access protection are activated once the remote connection has been terminated.
- The modified network parameters are activated immediately.

Configuring TS adapter MPI

Introduction

The TS Adapter MPI can be configured in both direct connection mode and via an existing remote connection in modem connection mode.

The following describes the method for assigning parameters in direct connection.

Requirement

A TS Adapter MPI is connected to your computer and the folder "TS adapter" is displayed in the project tree under "Online access".

Procedure

To assign the parameters for the TS Adapter MPI im Direktanschluss please proceed as follows:

1. In the project tree, double-click on the "TS adapter" folder under "Online access".
2. Select the command "Assign TS Adapter MPI parameters". The "Assign TS Adapter MPI parameters" dialog opens.
3. Set the desired parameters in the individual tabs of the dialog.
4. Confirm your entries with "OK".

Result

The configured parameters are saved in the non-volatile memory of the TS adapter MPI. Parameter assignment is then complete.

Restoring default parameter assignment for TS adapter MPI

Introduction

You can restore the default, factory state parameters of the TS Adapter MPI.

Requirement

A TS Adapter MPI is connected to your computer and is displayed in the project tree under "Online access" in the "TeleService" folder.

Procedure

Proceed as follows to restore default parameters for the TS Adapter MPI:

1. Open the "TeleService" folder in project tree.
2. Double-click the "TS Adapter MPI" folder.
3. Select the command "Assign TS Adapter MPI parameters". The "Assign TS Adapter MPI parameters" dialog opens.
4. Click the "Reset" button under "General".
5. Confirm your entries with "OK".

Result

The TS Adapter MPI default parameters set on delivery are restored.

See also

TS adapter MPI configuration options (Page 3105)

Exporting adapter parameters

Introduction

You can export the configuration of a TS Adapter MPI to an external file. The configuration saved in this file can be imported in turn into any number of TS Adapter MPI.

This can for example be useful if you want to assign identical parameters to multipleTS Adapter MPI or if you want save, document or distribute the parameter set.

Requirement

A TS Adapter MPI is connected to your computer and is displayed in the project tree under "Online access" in the "TeleService" folder.

Procedure

To export the adapter parameters of a TS Adapter MPI:

1. Open the "TeleService" folder in project tree.
2. Double-click the "TS Adapter MPI" folder.
3. Select the command "Assign TS Adapter MPI parameters". The "Assign TS Adapter MPI parameters" dialog opens.
4. Click the "Export" button.
5. A window will open in which you can select the file to which you wish to export the configuration of the TS Adapter MPI.
6. Confirm with "Save".

Result

The parameters of the TS Adapter MPI are saved in the specified file (*.tap). The export of the adapter parameters is now complete.

Importing adapter parameters

Introduction

You can import the configuration of a TS Adapter MPI from a previously created export file (*.tap).

The configuration saved in this file can be imported into any number of TS Adapter. This can for example be useful if you want to assign identical parameters to multiple TS Adapter MPI.

You can import parameters locally in direct connection mode or via an existing remote connection in modem connection mode.

Requirement

A TS Adapter MPI is connected to your computer and is displayed in the project tree under "Online access" in the "TeleService" folder.

Procedure

To import the adapter parameters of a TS Adapter MPI:

1. Open the "TeleService" folder in project tree.
2. Double-click the "TS Adapter MPI" folder.
3. Select the command "Assign TS Adapter MPI parameters". The "Assign TS Adapter MPI parameters" dialog opens.
4. Click the "Import" button.
5. A dialog will open in which you can select the file to which you wish to import the configuration of the TS Adapter MPI.
6. Confirm the next dialog with "Yes".

Result

The parameters selected are saved in the non-volatile memory of the TS Adapter MPI. Adapter parameter import is then complete.

12.12.3.4 TS adapter IE

Short description of the TS adapter IE

TS Adapter IE

The designation "TS Adapter IE" is a collective term for allTS Adapter with an Ethernet port.

The TS Adapter IE comes in the following versions:

- as TS Adapter IE Standard
- as TS Adapter IE Basic

The tables below provide a short description of the functionalities. For detailed information on your TS Adapter, please refer to the documentation supplied with it.

TS Adapter IE Standard:
Direct connection over Industrial Ethernet (IE). Replaceable firmware. Modem integrated or external. The TS Adapter IE cannot automatically switch between modems like the TS Adapter II. Parameters are assigned via a Web interface.
There are 2 variants:
• With internal analog modem. An external modem can also be connected to the RS232 port.
• With internal ISDN adapter. An external modem can also be connected to the RS232 port.

TS Adapter IE Basic:
Direct connection over Industrial Ethernet (IE). Replaceable firmware. Plug-in modules. Parameters are assigned via a Web interface.
There are 4 variants:
• TS Adapter IE Basic MODEM: Basic device TS Adapter IE Basic with TS Module MODEM for operation on the analog telephone network.
• TS Adapter IE Basic ISDN: Basic device TS Adapter IE Basic with TS Module ISDN for operation on ISDN telephone systems.
• TS Adapter IE Basic GSM: Basic device TS Adapter IE Basic with TS Module GSM for operation on the GSM radio network.
• TS Adapter IE Basic RS232: Basic device TS Adapter IE Basic with TS Module RS232 for connecting an external modem.

Use of the designation "TS Adapter"

"TS Adapter" is used in the TeleService online help as a general designation for all versions. The relevant product designation is listed beside information which only applies to a specific version of a TS Adapter, e.g. "TS Adapter I", "TS Adapter II", "TS Adapter IE Standard" or "TS Adapter IE Basic".

How the TS adapter IE works

How the TS Adapter IE works

The TS Adapter IE connects the telephone network or the serial port of a modem with the Industrial Ethernet of your automation system.

The TS Adapter IE has a non-volatile memory. Parameters for the following functions are stored in this memory:

- The mode of the modem used
- The serial port to the modem
- Access protection

Default parameter assignment

The TS Adapter IE comes with default parameter assignment. The parameters can be set and saved to the non-volatile memory of the TS adapter in a parameter assignment session.

Note

For more detailed information on the configuration of your TS Adapter, please refer to the documentation supplied with it.

Connection Types

Connection types of the TS Adapter IE Basic

The following diagrams show the connection types possible with the TS Adapter IE Basic.

Direct connection

In the direct connection to the PG/PC, you can set the TS Adapter IE Basic through Ethernet.

Note

The operation of the TS Adapter IE Basic without a TS module is not permitted.

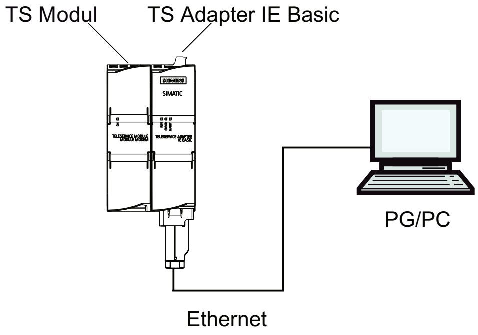


Figure 12-2 Direct connection

Connection to the telephone network

In order to have a direct connection to the telephone network, you must connect the TS Adapter IE Basic together with one of the following TS modules:

- TS Module Modem
- TS Module ISDN

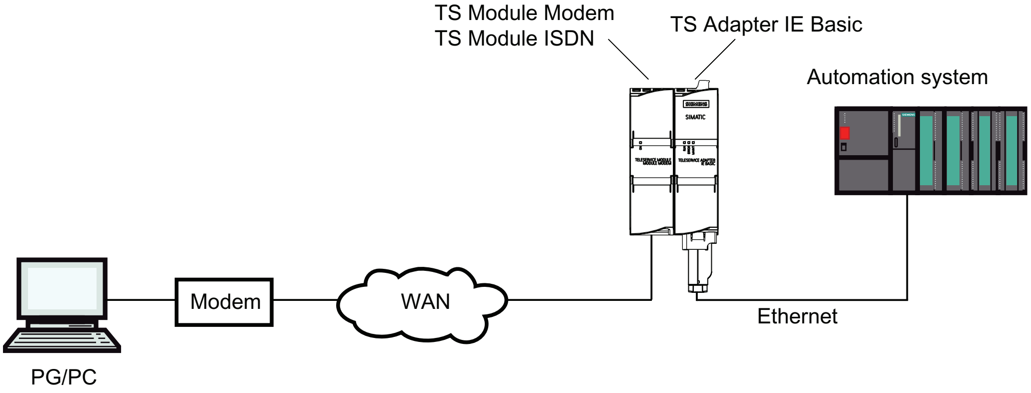


Figure 12-3 Direct connection to the telephone network

More information about the TS modules can be found in the *TS Adapter modular* manual.

Connection to the GSM network

In order to connect to the GSM network, you must operate the TS Adapter IE Basic together with this TS module:

- TS Module GSM

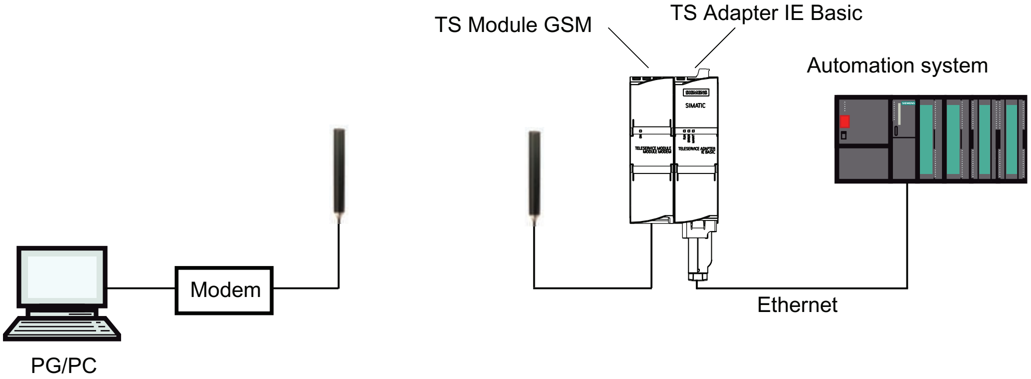


Figure 12-4 Connection to the GSM network

More information about the TS modules can be found in the *TS Adapter modular* manual.

Connection to the telephone network through an external modem

For the connection to an external modem, you must operate the TS Adapter IE Basic together with this module:

- TS Module RS232

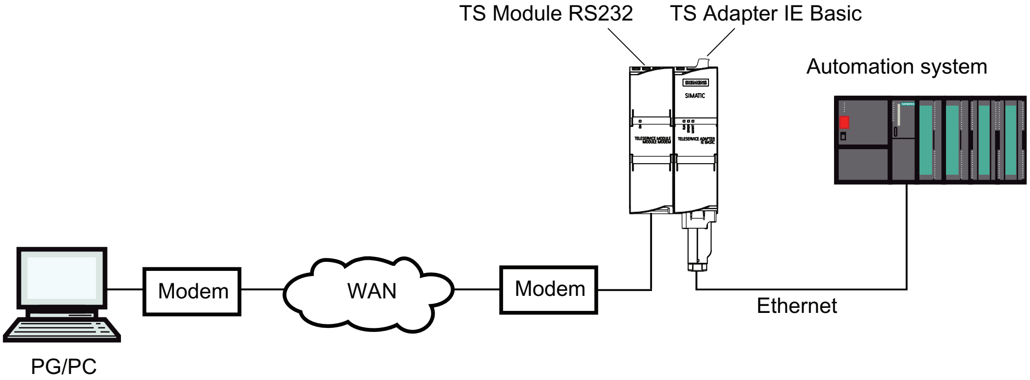


Figure 12-5 Connection to an external modem

More information about the TS modules can be found in the *TS Adapter modular* manual.

TS Adapter IE parameter assignment options

Useful information for configuring the TS Adapter IE

The TS Adapter IE is configured via a Web interface.

Web help associated with the parameter assignment interface is made available for the configuring the TS Adapter IE.

The following parameter assignment options are available:

- Reconfiguration
- Restoring default parameter assignment
- Importing adapter parameters
- Exporting adapter parameters

Note

Parameter assignment

Configure your TS Adapter in accordance with the documentation supplied with the TS Adapter. It will detail the exact procedure for parameter assignments.

Parameter assignment for TS Adapter IE

<Introduction

The TS Adapter IE can be configured in both direct connection mode and via an existing remote connection in modem connection mode.

The following describes the process of assigning parameters in direct connection.

Specific details for assigning parameters of the TS adapter IE can be obtained from the TS adapter IE documentation.

Requirement

There is a LAN connection to your TS Adapter IE Basic.

The TS Adapter IE Basic is connected to the power supply.

Procedure

To assign the parameters for the TS Adapter IE please proceed as follows::

1. In the project tree of the TIA portal, open the "Online access" folder.
2. Double-click on the Ethernet port of your computer.
3. Double-click on the "Display accessible nodes" command. The TS adapter IE is then displayed.
4. Double-click on the <TS adapter IE> folder and then on "Online and diagnostic", and assign the desired IP address to the TS adapter IE in the following dialogs. Please note that the IP address of the PG/PC interface card is located in the same subnet as the IP address that you issue for the TS adapter IE.
5. Update the view in the project tree for the "Accessible nodes", so that the TS adapter IE is displayed with the newly allocated IP address.
6. Open the folder <TS adapter IE> in the node list.
7. Double-click the command "Assign TS adapter IE parameters". The allocated web-interface opens for assigning the TS adapters IE parameter.
8. Complete the "logon" for the web interface.
9. Set the desired parameters in the individual tabs of the dialog.
10. Confirm your entries with "Save settings".

Result

The configured parameters are saved in the non-volatile memory of the TS adapter IE. Parameter assignment is then complete.

12.12.4 Access protection for TeleService and the TS Adapter

12.12.4.1 Access protection information

Introduction

When you assign the parameters for your TS adapter, you can restrict access to the parameters of the TS adapter and access to remote systems.

Scope of access protection

Access protection only exists for remote connections; TS adapter parameter assignment can be accessed at any time in direct connection mode.

Access protection also exists in direct connection for the TS adapter IE.

Access protection information

The TS Adapter MPI is not delivered with access protection activated. There is a default password for the TS adapter IE.

The first user who assigns the parameters for this adapter can therefore activate access protection by defining the password for a user and/or a callback number.

This is a multi-level access protection with several users, each with or without administrator rights. For the TS adapter MPI there is only one adminstrator and no more than two users.

For a modem connection, only an administrator can define the two users and change and, if necessary, delete their settings. Those logged in as users can only change their own passwords and their own callback numbers. However, with the TS adapter MP you can access the process of assigning parameters of the TS adapter in direct connection, without restriction.

Advantages

Access protection offers the following advantages:

- Unauthorized access by persons outside the system is almost impossible.
- The plant operator bears most of the telephone costs.

12.12.4.2 TeleService callback options

Callback variants

The costs of a telephone connection are normally borne by the caller who sets up the TeleService session.

TeleService can, however, be used so that after a short initial connection the modem connection is established again in the opposite direction, in other words initiated by the TS Adapter (callback). In this case, the plant operator bears the costs of the callback.

There are two callback variants in TeleService:

1. Callback to a number specified during connection establishment.
2. Callback to a number stored on the TS Adapter.

12.12.4.3 Levels of protection

Introduction

You can set up one of two possible levels of access protection for TeleService access to the TS Adapter. Different options are available with each protection level.

Access protection options

Access protection level 1:

The TS Adapter is protected by the user name and password. You can access the TS Adapter via any telephone line and specify any callback number during connection establishment.

Access protection level 2:

The TS Adapter is protected by the user name, password, and the callback number. You can only access the TS Adapter from one telephone connection per user.

The table below sets out the above conditions for the various protection levels:

Level of access protection	Administrator/User password	Callback number
1	enter	do not enter
2	enter	enter

Logging on to TS Adapter

When you log on to the TS Adapter and after you have set up access protection, enter your user name, the corresponding password and, if desired, a callback number:

Level of access protection	Administrator/User password	Callback number
1	enter	do not enter or enter any callback number
2	enter	do not enter

If you have entered a callback number during connection establishment (access protection level 1) or stored a callback number in the TS Adapter (access protection level 2), the modem connection will be terminated and the TS Adapter will call back the given number.

12.12.4.4 Setting up access protection and callback number for the TS adapter

Introduction

During the parameter assignment for the TS adapter MPI in TeleService, you can set up access protection and a callback number for the parameter assignment of the adapter and connection to the remote system. The following describes the parameter assignment for a TS adapter MPI. The parameter assignment of a TS adapter IE is carried out in analog. The specific method is described in the web help of this adapter.

Requirement

A TS Adapter MPI is connected to your computer and is displayed in the project tree under "Accessible nodes".

Procedure

To set up access protection for the TS adapter, proceed as follows:

1. Click on the command "Assign TS Adapter MPI parameters" in the project tree.
2. Open the "Access security" tab.
3. Enter a password for your user name and/or number that you want the modem to call back following logon.
   – If you are an administrator, you can change all the settings for administrators and users, and delete or create users.
   – If you are logged on as a user, you can only change your own settings (password and callback number).
4. Confirm all entries before exiting the dialog with "OK".
5. Click the "Yes" button to confirm the following query.

Result

The parameter assignment for the access protection and the callback number is saved in the non-volatile memory of the TS adapter MPI.

Note

Important points to note when setting up access protection:

- The settings in the "Modem" tab must correspond to the conditions at the plant if callback functionality is to be guaranteed.
- Entering an incorrect callback number in the role of "ADMIN" user will mean you are no longer able to access the TS Adapter MPI over a remote connection!
- Test the callback number before you enter it as the "ADMIN" user by calling the given callback number during connection establishment (access protection level 1).

12.12.4.5 Complete a callback in TeleService

Callback options

Two different callback variants can be set up in TeleService.

The following callback options are available:

- Callback to a number specified during connection establishment.
- Callback to a number stored on the TS Adapter

Callback to a number specified during connection establishment

1. In the project tree of the TIA Portal, open the "Online access" folder.
2. Then click the "TeleService" folder contained within.
3. Double-click the "Set up/close remote connection" entry. The "Set up remote connection to the remote system" dialog opens.
4. Select the adapter type used in the "TS Adapter" drop down list.
5. Select the telephone network under "Network" if it is not already selected.
6. Select the modem you are using under "Local Settings".
7. Enter the phone number to be dialed in the appropriate box or open the phone book by clicking on the button behind it and take the desired phone number from the phone book.
8. Enter your user name and associated password of the TS adapter.
9. If you want a "Connection setup with callback", select the appropriate option button.
10. Click the "Dial" button to establish the desired remote connection. This button only becomes active when you have entered all the parameters needed establishing a remote connection. Any remote connection is displayed under "Status".
11. Enter the desired callback number in the dialog that follows.

Result

The remote connection to the desired system is made with callback.

The connected system is shown with the corresponding icon in the project tree.

Note

This procedure is useful if the costs of the modem connection are to be borne by the plant operator and if the actual callback number is not fixed, i.e. callback is not always to the same receiver. It is particularly useful for mobile users.

Callback to a number stored on the TS Adapter

1. Assign the parameters for the desired callback number in the TS adapter.
2. Establish a connection to the TS adapter as described above, and observe the following features:
   – Enter the user name and password for which the callback number parameters are assigned in the TS adapter.
   – The optional field "Establish a connection with callback" does not have to be selected, since the callback number is already known by the TS adapter.

Result

Callback to a number stored on the TS adapter has been established. If a remote connection is established, the callback occurs from the remote system.

Note

This procedure offers the highest level of access protection. However, it does pose a risk: if the callback number stored on the TS Adapter is not correct, it will no longer be possible to access the TS Adapter over a modem connection. The device can in such a case only be put back into operation by changing the parameter settings on site.

12.12.5 Establishing a remote connection to a remote plant

12.12.5.1 Establishing a remote connection

Introduction for establishing a remote connection

A remote connection is established when you use TeleService to dial into a remote system via a telephone network. The programming device/personal computer is connected to the telephone network with TeleService via a modem. At the other end, the automation system is connected to the telephone line via a configured TS Adapter and a modem.

Requirements

A local modem is installed and configured.

The TS adapter is located in the remote system.

A remote modem is installed and parameters have been assigned.

Proceed as follows:

1. In the project tree of the TIA Portal, open the "Online access" folder.
2. Then click the "TeleService" folder contained within.
3. Double-click the "Set up/close remote connection" entry. The "Set up remote connection to the remote system" dialog opens.
4. Select the adapter type used in the "TS Adapter" drop down list.
5. Select the telephone network under "Network".
6. Select the modem you are using under "Local Settings".
7. Enter the phone number to be dialed in the appropriate box or open the phone book by clicking on the button behind it and take the desired phone number from the phone book.
8. Enter the your user name and associated password.
9. If you want a "Connection setup with callback", select the appropriate option button.
10. Click the "Dial" button to establish the desired remote connection. This button only becomes active when you have entered all the parameters needed establishing a remote connection. Any remote connection is displayed under "Status".

Result

The remote connection to the desired plant is established. In "Status" the progress of establishing the connection is displayed: First "Select", then "Authenticate".

The dialog closes once the remote connection is established. The message appears in the TIA portal status line: "Remote connection is established". You can now use the remote connection with TIA portal and communicate with the automation system.

If the connection cannot be established ...

If the connection cannot be established, try to find the cause using the "Troubleshooting" information.

Terminating the connection

Once you have finished editing the remote system, exit the remote connection in the project tree by double-clicking on the entry "Establish/disconnect remote connection".

By exiting the TIA portal you are also exiting the remote connection.

12.12.5.2 Terminating a remote connection

To disconnect an active remote connection, proceed as follows:

1. Double-click the "Set up/close remote connection" entry.

Result

The connection will be terminated immediately.

"Offline" status will be displayed in the status row again once the remote connection has been terminated.

Note

You should go offline in the TIA Portal before you terminate the remote connection.

12.12.5.3 Checklist for troubleshooting the modem

Introduction

The following list should help you establish the potential cause of any problems with the modem. The help topics below set out how and in which dialogs you define the relevant settings.

Modem connection cannot be established:

- Check the cabling and the connections.
- Have you set the correct dialing mode (tone/pulse)?
- If your modem does not react after several attempts to dial, a dial disable function may be active. Familiarize yourself with dial disable on your modem.
- Are you operating your modem on a main telephone line or on an extension line? Configure the properties and dialing parameters of the modem accordingly.
- Enable the log file option in the advanced properties. The next attempt to establish a connection will then be recorded in a file in the Windows directory.
- Ensure that the ISDN TAs used work with the same B and D channel protocol.

The modem connection is terminated:

- Metering pulses can have a negative affect on a connection. Have the pulses deactivated by your telephone company.
- Set fixed monitoring times.
- Deactivate the option that terminates an existing connection automatically after a specified time without data transfer (idle).
- Ensure that you have activated the RTS/CTS protocol for data flow control.

12.12.6 Working with the phone book

12.12.6.1 Basics on working with the phone book

Working with the phone book

You have the following options when working with a phone book:

- Open phone book
- Save phone book
- Import phone book data
- Export phone book data
- Printing the phone book
- Use phone book data to establish a remote connection.

You can implement these functions simply and easily using the buttons in the toolbar.

Note

Access to phone books

The phonebook is user specific in TeleService. However, it is not possible to access the global telephone book with more than one instance of the TIA portal at the same time.

See also

Open phone book (Page 3124)

Saving phone book (Page 3124)

Importing phone book data (Page 3124)

Exporting phone book data (Page 3125)

Printing the phone book (Page 3126)

12.12.6.2 Structure of the phone book

Introduction

A global telephone book in TeleService is used to manage the data you require for establishing a remote connection. Once you have created the connection data and saved it in the telephone book, you can access it each time you want to establish a remote connection.

Structure of the phone book

The integrated global telephone book in TeleService contains the following columns:

Column name	Meaning
Plant name	Enter a name for your plant
Adapter type	Select the TS Adapter type used in the drop down list.
Area code	Enter the area code.
Telephone number	Enter the telephone number to be dialed for the remote connection.
Country	Enter the country code.
User name	Enter the user name you have logged on under.
Password	Enter the password for this user name.
Group	Enter the group if you have carried out grouping.
Company	Enter the company to be called.
Department	Enter the company department to be called.
Street	Enter the street.
Town/City	Enter the town or city to be called.
Comment	Enter a comment if required.

12.12.6.3 Symbols in the phone book

Meaning of the TeleService icons

The following table shows the meaning of the TeleService icons:

Symbol	Meaning
📖	Open the global telephone book.
⇠	Import a telephone book.
⇢	Exports a phone book.
☎↑	Establishes a remote connection.
☎↓	Closes the active remote connection.
☎⇅	Establishing or closing a remote connection.

12.12.6.4 Manage phone book

Open phone book

Opening phone books

To open the phone book, proceed as follows:

1. In the project tree, double-click on the "Phone book" folder under "Online access" > "TeleService".
2. The phone book opens so that you can enter or edit the desired plant data.

Saving phone book

Saving phone books

The global phonebook is saved automatically when you exit the phone book editor or when leave the TIA Portal.

Importing phone book data

Introduction

It is possible to import the phone book data from an external file. The configuration saved in this file can be imported in turn into any number of TS Adapter MPIs.

Requirement

You have already created an import-capable phone book file under TeleService.

Procedure

Proceed as follows to import the phone book data:

1. Open the "TeleService" folder in project tree.
2. Double-click on the "Phone book" folder.
3. Click the "Import" button in the toolbar.
4. Confirm the prompt asking if you want to save the current state of the phone book with "Yes" when applicable, and specify the location for storing the phone book in the dialog that follows.
5. If you do not want save the current state of the phone book, answer the prompt with "No". In the subsequent dialog, select the phone book file to which in the current phone book should be stored.
6. Close the dialog box with "OK".

Result

The imported phone book data is displayed in the global phone book.

Exporting phone book data

Introduction

It is possible to export the phone book data to an external file. The parameter assignment saved in this file can be imported in turn into any number of TS adapter.

Requirement

You have already created a phone book under TeleService with the corresponding plant data.

Procedure

Proceed as follows to export the phone book data:

1. Open the "TeleService" folder in project tree.
2. Double-click on the "Phone book" folder.
3. Click the "Export" button in the toolbar.
4. In the next dialog, select where the current phone book is to be exported.
5. Close the dialog box with "OK".

Result

The exported phone book data are saved in the specified export file.

Printing the phone book

Printing phone books

You can print out all or some of the data in a phone book.

Proceed as follows:

1. Open the telephone book.
2. Select the **Phone book > Print** menu command or click on the appropriate button in the toolbar. The "Print" dialog will open.
3. Specify whether you wish to print the complete phone book or just part of it and set all other options.
4. Start the print job with "OK".

Result

The phone book data is printed on the default printer. If the printout is more than one page long, an identifier is printed after the page number at the bottom right corner of the page to indicate that there is another page. The last page does not have this symbol, indicating that no more pages are to follow.

12.12.7 CPU controlled TeleService remote connections

12.12.7.1 Overview of CPU controlled remote connections

Introduction

TeleService offers a range of options for establishing remote connections; these differ according to the CPU used. The initiative for establishing a connection starts from the CPU. The communication instructions given below are used for the individual connection options.

Connection establishment with S7-300/400 CPUs

The following communication instructions are available:

- Communications instruction "PG_DIAL": Establish remote connection to programming device/PC
- Communications instruction "SMS_SEND": Send text message (SMS)
- Communications instruction "AS_DIAL": Establish remote connection to AS
- Communications instruction "AS_MAIL": Transfer email

Connection establishment with S7-1200 CPUs

The following communication instruction is available:

- Communications instruction "TM_MAIL": Transfer email

Note

Description of individual communication instructions

More detailed information on the available communication instructions can be found in the information system of the TIA portal in the directory "References > Communication > TeleService".

12.12.7.2 Establishing a connection from and to remote systens (PG-AS-remote coupling)

Remote plant access to a programming device/personal computer

Introduction

You can establish a remote connection to and communicate with a remote system using the application TeleService and a TS Adapter MPI. The initiative for establishing the remote connection comes from the programming device/personal computer.

However, events which require rapid intervention often occur at a remote system. In such cases, the automation system can initiate a remote connection to a programming device/personal computer if an asynchronous event occurs.

The graphic below shows the components which are required for establishing a connection from a plant to a programming device/personal computer.

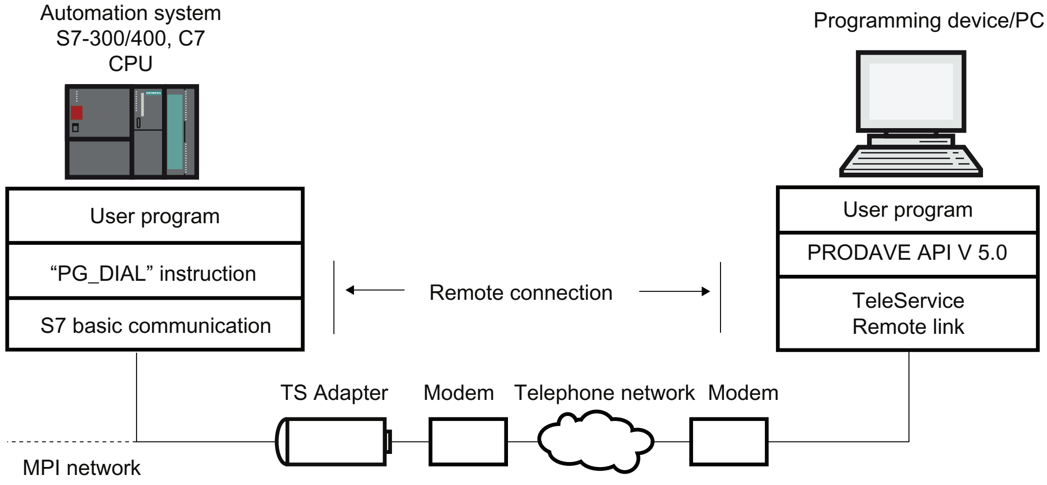


Figure 12-6 How the "PG_DIAL" communications instruction works

Requirements for establishing a connection

Introduction

Certain hardware and software requirements must be fulfilled if a remote system is to establish a remote connection to a programming device/personal computer. These requirements are detailed below.

Hardware requirements:

The only hardware required for establishing a remote connection from a remote system to a programming device/personal computer is that needed for accessing the remote system from the programming device/personal computer.

Your user program calls the communication instruction "PG_DIAL" to establish the connection. This can only be executed on an S7-300 or S7-400 CPU on which S7 basic communication is implemented.

A TS Adapter I , version 5.0 or later, or a TS Adapter II must be used.

Software requirements at the plant end:

Communication instruction "PG_DIAL" is included in the TeleService scope of delivery and is installed when the TIA portal is installed. You will find the communication instructions installed in the "Communication > TeleService" folder of the block editor task card.

If a remote system is to establish a remote connection to a programming device/personal computer, the plant user program must call the "PG_DIAL" function block.

Software requirements for the programming device/personal computer

You require a software product in the programming device/personal computer which, with TeleService, waits for a call from a remote system, recognizes this call and informs your user program.

12.12.7.3 Data exchange between remote systems (AS-AS-remote coupling)

AS-AS remote link basics

Introduction

The AS-AS remote link allows two automation systems to exchange process data via the telephone network.

Requirement

Communication instruction "AS_DIAL" is available if you use a CPU from the S7-300/400 family.

Definition: Local and remote automation system

- The automation system from which the initiative to establish the remote connection originates is described as local.
- The automation system to which the remote connection is to be established is described as remote.

Data exchange over the AS-AS remote link

Data exchange is carried out using specific communication instructions for non-configured S7 connections. Use the communication instruction "AS_DIAL" to establish a remote connection to the automation system.

More detailed information on establishing the connection can be found in the information system in the directory "References > Communication > TeleService".

The following graphic shows the components required for establishing a connection from a local to a remote automation system.

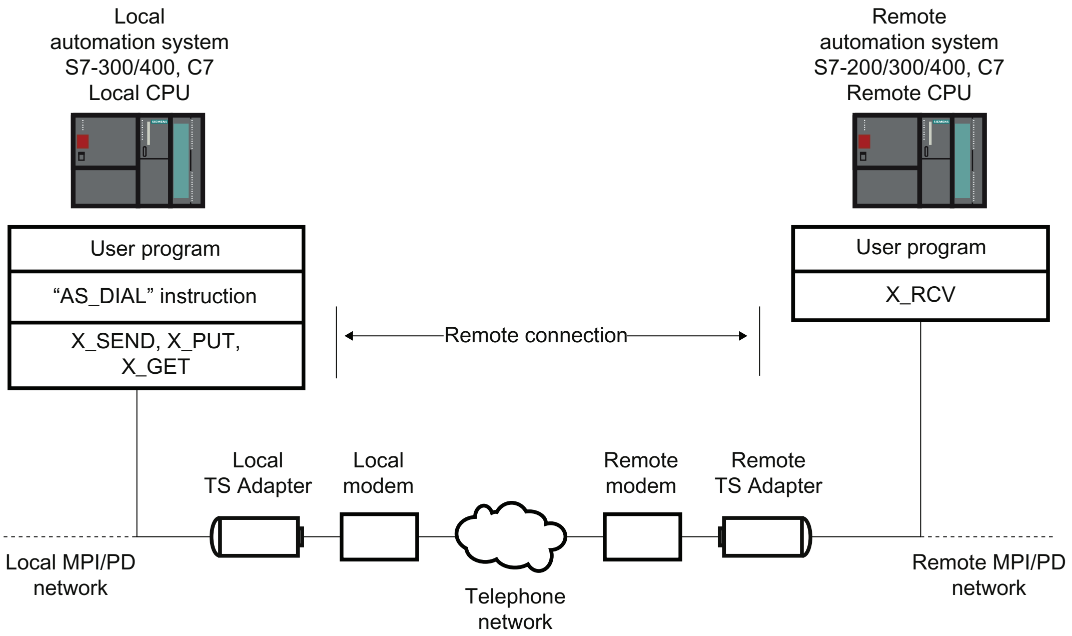


Figure 12-7 Data exchange over the AS-AS remote link

Hardware and software requirements for AS-AS remote link

Introduction

Certain hardware and software requirements must be fulfilled before a local automation system can establish a remote connection to a remote automation system. These requirements are detailed below.

Hardware requirements

The only hardware you need for transferring process data from a local to a remote automation system is that also needed for accessing the respective automation system from the programming device/personal computer.

To establish and terminate the remote connection, the TIA Portal user program of the local CPU calls a communication instruction. This communication instruction can be executed on an S7-300/400 CPU or a C7 CPU. The communication instruction requires S7 basic communication to be implemented on the CPU. The remote CPU must also support S7 basic communication.

A TS Adapter I, version V5.1 or later, or a TS Adapter II must be used.

Software requirements

The "AS_DIAL" communication instruction is included in the product package of TeleService, and is integrated into the library of the TIA Portal during the installation in the communication instructions folder of the Task Card under TeleService. In order to establish and terminate a remote connection to a remote automation system from a local automation system, call the communication instruction "AS_DIAL" in the TIA Portal user program of the local CPU.

AS-AS remote link

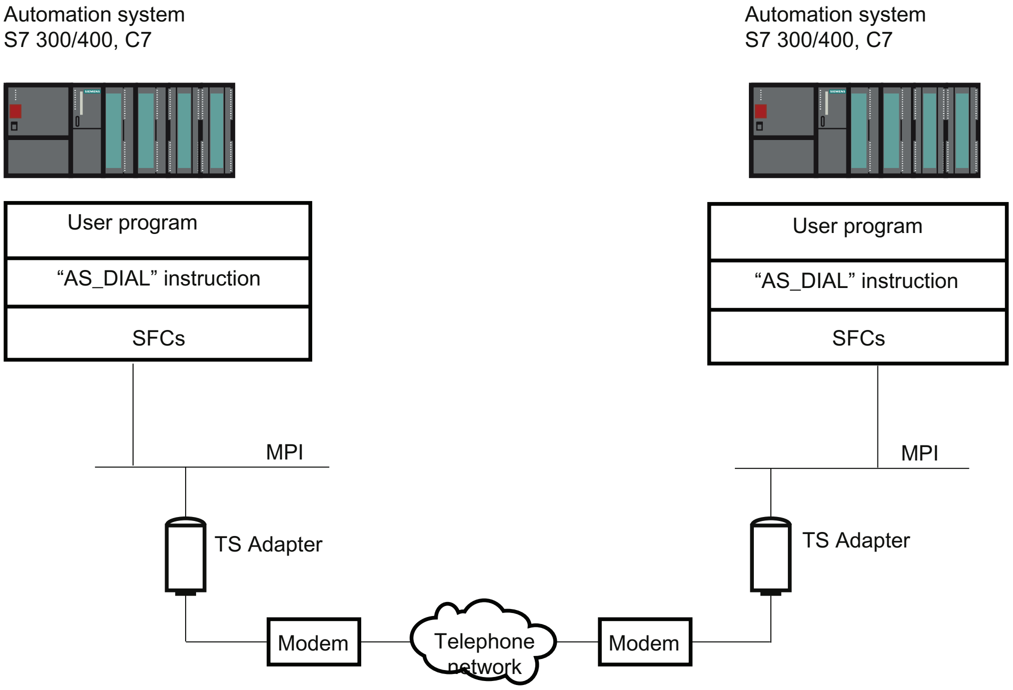


Figure 12-8 Hardware and software requirements for AS-AS remote link

12.12.7.4 Send SMS from a system

Requirements for sending an SMS

Introduction

Certain hardware and software requirements must be fulfilled before a plant can send an SMS. These requirements are detailed below.

Hardware requirements

To send an SMS from a plant, you will require a GSM radio modem and a TS Adapter MPI.

A TS Adapter I, version V5.2 or later, or a TS Adapter II must be used.

Software requirements at the plant end

The "SMS_SEND" communication instruction is included in the product package of TeleService, and is integrated into the library of the TIA Portal during the installation in the communication instructions folder of the Task Card under TeleService. If a plant is to send an SMS, the user program of the plant must call the communication instruction "SMS_SEND".

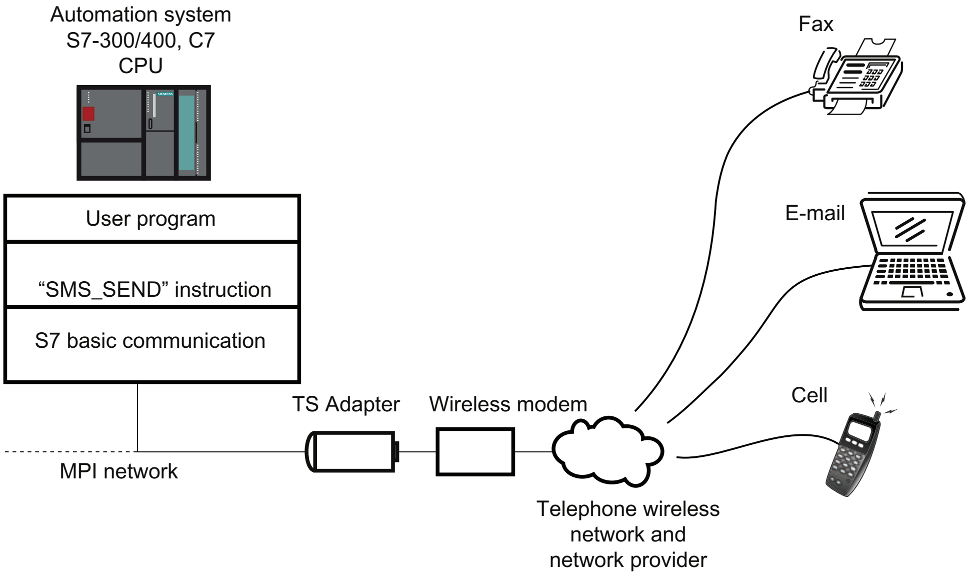


Figure 12-9 How the "SMS_SEND" communication instruction works

12.12.7.5 Send an email from a system

Requirements for sending e-mails

Introduction

The following hardware and software requirements must be fulfilled if a plant is to send an e-mail:

Hardware requirements

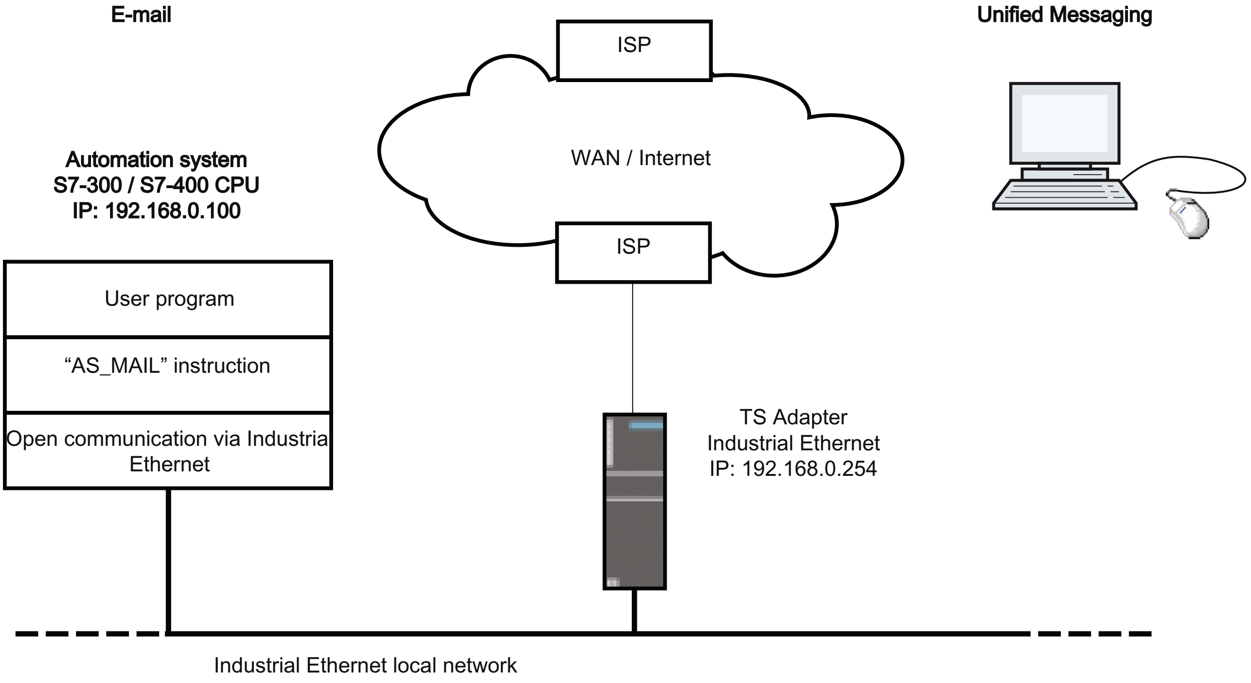
To send e-mail from a plant, you will need a TS Adapter IE and a CPU 31x-2 PN/DP, firmware version V2.5 or higher, or a CPU 41x-3 PN/DP.

Software requirements at the plant end

The "AS_Mail" communication instruction is included in the product package of TeleService, and is integrated into the library of the TIA Portal during the installation in the communication instructions folder of the Task Card under TeleService.

If a plant is to send an e-mail, the user program of the plant must call the communication instruction "AS_MAIL". This uses the Simple Mail Transfer Protocol (SMTP) to send e-mails from a CPU to a mail server.

You will also need the communication instructions described below from the TIA Portal standard library.



The "Use gateway/router" property must also be set for the Ethernet interface in the configuration of the CPU on which the communication instruction "AS_MAIL" runs. (See: Hardware configuration > PROFINET IO (PN-IO) > General > Properties > Parameters). The IP address of the Ethernet interface of the TS Adapter IE should be specified as the "Address".

Required communication instructions

Communication instruction "AS_MAIL" requires the following communication instructions from the "IEC Function Blocks" folder of the TIA Portal standard library:

- Communications instruction "EQ_STRNG"
- Communications instruction "FIND"
- Communications instruction "INSERT"
- Communications instruction "LEFT"
- Communications instruction "LEN" and
- Communications instruction "RIGHT"

Copy these communication instructions from the standard library of the TIA Portal to your project and your CPU.

Note

You can find further information in the task card in the Communication instructions folder under TeleService.

12.12.8 Troubleshooting

12.12.8.1 General information on troubleshooting

Introduction

The information below should help you establish and eliminate the causes of any modem problems.

1. Activate "Record log file for data traffic between PG/PC and modem". The entries in this file can provide valuable information for determining the cause of errors.
2. Switch on the loudspeaker on your local modem. Select a volume loud enough to be clearly heard.

   You can then hear whether:

   – there is a dial tone at the connection

   – the modem called is busy

   – the modem called accepts the call.

Common modem problems

Modem connection problems are among the most common modem problems:

- Modem connection is not established
- Modem connection is interrupted

The topics below contain tables detailing possible causes and providing information on eliminating the error in question.

See also

Remote connection to the TS adapter is not established (Page 3136)

Remote connection from the TS adapter is not established (Page 3137)

Modem connection is interrupted (Page 3138)

Modem alarms (Page 3139)

Recording a log file for the modem (Page 3135)

12.12.8.2 Recording a log file for the modem

Introduction

It is advisable to record a log file as this makes it easier to find the causes of faults in a modem.

Procedure:

Proceed as follows:

1. Activate the properties dialog of the modem used via the "Phone and modem options" option in the Control Panel.
2. Check the settings of the "Log" option in the "Diagnostics" tab and if necessary change the log file settings so that the file is recorded.

Result:

Activity between the programming device/personal computer and the modem are entered in the log file. If there are problems establishing the connection, you can evaluate the data recorded in the log file to determine the cause of the error.

12.12.8.3 Remote connection to the TS adapter is not established

Remote connection to TS Adapter is not established

The table below sets out possible causes and how to eliminate them if no remote connection to the TS Adapter can be established.

Possible cause	Check/Remedy
Cabling faulty	• Are all the connecting cables connected correctly? • Are the connectors loose?
Dial parameters for main telephone line and extension incorrectly set	• Are the modem properties and dial parameters set for your modem correct for the phone connection (main telephone line or extension)? • Do not specify a dial-out code in the "Dial parameters" dialog if you operate your modem on a local loop (main telephone line). The fields for the dial-out code for local calls and long-distance calls must be empty.
Dialing mode incorrectly set	• Is the correct dialing mode (tone/pulse) set in the dialog for the dial parameters of your modem? • Use a connected telephone to check the connection on which you want to operate the modem. You should hear crackling noises on the telephone during pulse dialing and tones of varying pitches during tone dialing. Set the corresponding dialing mode in the modem dial parameters.
Dial disable active	• The dial disable function is a country-specific modem property which, depending on the modem, becomes effective after one or more unsuccessful attempts to establish a connection. • If your modem still does not respond after several attempts to dial, the dial disable function may be active. Characters are still sent to the modem after the dial command but the modem does not start the dialing process. The driver receives a general error message. • Refer to the modem documentation for information on how the dial disable function is implemented for your modem. • Create a log file (Page 3135) (modemlog.txt) in which the activities between the programming device/personal computer and modem are recorded. Then check whether the file contains an entry caused by dial disable (e.g. DELAYED).
Phone connection defective or busy	• Connect a phone and check whether a dial tone can be heard on this connection. • Any analog phone connected on the same connection must be hung up. You cannot establish an additional modem connection on this connection if there is an existing phone connection.
Serial parameters set incorrectly	• Are the correct values entered in the "Settings" tab for modem properties (8 data bits, no parity, 1 stop bit)? Is the correct COM interface set in the "General" tab for the modem properties?

Possible cause	Check/Remedy
Initialization string of the TS Adapter is not suitable for the modem.	• Familiarize yourself with the modem initialization string requirements (Page 3099) and set the string accordingly. Procedure for configuring the TS Adapter IE (Page 3110)
Settings for error correction between the modem at the TS Adapter and the modem at the PC/programming device are not compatible.	• Adapt the modem settings. Useful information on configuring the TS Adapter MPI (Page 3105) Restoring the default parameter assignment of a TS Adapter MPI (Page 3107) Procedure for configuring the TS Adapter IE (Page 3106)

12.12.8.4 Remote connection from the TS adapter is not established

No callback from TS Adapter

The table below sets out possible causes and how to eliminate them if there is no callback from the TS Adapter.

Possible cause	Check/Remedy
Errors in the location or call settings in the TS Adapter	Check the TS Adapter parameter assignment: • Are the dialing mode and dial-out code set correctly for your phone connection? • Does the modem at the TS Adapter support the characters configured for the dial-out code? • Is "Wait for dial tone before dialing" deactivated for an extension?
Initialization of modem insufficient	Check the string for modem initialization: • The modem may require a further initialization in order to establish a remote connection. • Properties of the modem initialization string for the TS Adapter MPI
Callback number is incorrect	Check the configuration of the callback number you assigned.

No call from TS Adapter MPI

The table below sets out possible causes and and how to remedy them if there is no call from the TS Adapter MPI.

Possible cause	Check/Remedy
Phone number is incorrect	Is the required number being transferred to the communication instruction "PG_DIAL" ?
TS Adapter MPI parameter assignment incorrect	Check the TS Adapter MPI parameter assignment: • Are the dialing mode and dial-out code set correctly for your phone connection? • Does the modem at the TS Adapter MPI support the characters configured for the dial-out code? • Is "Wait for dial tone before dialing" deactivated for an extension?

12.12.8.5 Modem connection is interrupted

Modem connection is interrupted

The table below sets out possible causes and and how to remedy them if the modem connection is interrupted.

Possible cause:	Check / Remedy:
Metering pulses in the line	• Metering pulses will be generated if you have applied to the phone company for a metering clock. This may mean that the modem no longer recognizes the carrier signal and switches off. • Set a longer waiting or switch-off time at the modem. • Have the metering pulse deactivated by the phone company.
Shielding	• Are the connection cables used shielded sufficiently? • Make sure that the modem cables do not run next to power cables and that they are as far as possible from power supply units and monitors.
Protocol timeout	• Set fixed monitoring times.
Automatic connection termination	• Deactivate the option that terminates an existing connection automatically after a specified time without data transfer ("Terminate after idle of ...").
Data flow control deactivated	• Click on the "Extended" button in the "Settings" tab of the modem properties and activate the following options in the dialog displayed (if available and not yet set): – Data flow control – Hardware (RTS/CTS) – Data compression – Error control
Initialization string of the TS Adapter is not suitable for the modem	• Set the modem initialization string in accordance with the following requirements: For further details, see: Requirements of the modem initialization string for TS Adapter MPI Options for the process of assigning parameters for the TS Adapter IE

See also

Initialization string requirements (Page 3099)

TS adapter MPI configuration options (Page 3105)

TS Adapter IE parameter assignment options (Page 3113)

12.12.8.6 Modem alarms

Information in the log file

The modem alarms are entered in a log file if you have activated the recording function.

The log file contains the following information:

Alarm:	Possible cause:	Remedy:
NO DIALTONE	A phone call may currently be being carried out on this line.	• Repeat the process once the phone call is over.
NO CARRIER	The device dialed is not ready, is not a modem or cannot establish a connection in the set operating mode.	• Check the numbers and the settings.
BUSY	The device dialed is busy.	• Try again later.
DELAYED: ...	Dial disable	• Refer to the modem documentation for information on how the dial disable function is implemented for your modem and if necessary remove it.

Index

"

"TM_MAIL" instruction,
"WWW" instruction,

*

*.bmp, 2005
*.emf, 2005
*.gif, 2005
*.ico, 2005
*.jpeg, 2005
*.jpg, 2005
*.tif, 2005
*.wmf, 2005

:

:add external graphic,
:add graphic to graphics library,
:P, 540

3

32-bit down counter, 599

A

Ability to read back connection parameters, 408
ABS, 1300, 1469, 1586
Absolute Addressing
  of a tag, 2118, 2403
Absolute value, 1300, 1469, 1586
Access protection, 2275
  Access protection level 1, 3116
  Access protection level 2, 3116
  Advantages, 3115
  Configuring, 2279
  Factory state, 3115
  User administration, 2274
  Validity, 3115
Access to operands, 724, 725, 726, 728, 729, 730, 731, 732, 733, 734, 735, 738, 739
Access to phone books, 3122
Accessible devices, 514, 2431, 2483
ACK, 2833
Acknowledge, 2161, 2365, 2368
  Alarm, 2835
  Error alarm, 2835
AcknowledgeAlarm, 2325
Acknowledgment, 2164
  Configuring, 2187, 2188, 2189
Acknowledgment concept
  Alarm with simple acknowledgment, 2165
  Alarm without acknowledgment, 2165
Acknowledgment model, 2165
ACOS, 1317, 1489, 1604
Acquisition cycle
  Area pointer, 2408
  Tag, 2136, 2142, 2154
Acquisition mode
  Tag, 2136
Actions
  Basics of redoing actions, 257
  Basics of undoing, 257
  Redoing, 260
  Undoing, 259
Activate
  Project language, 2727
ActivateCleanScreen, 2301
ActivatePreviousScreen, 2301
ActivateScreen, 2299
ActivateScreenByNumber, 2300
Active
  Area pointer, 2408
Adapting a project
  For a different HMI device, 2784
Add, 1289, 1458
  Objects to the group, 2012
ADD, 1289, 1458
Add empty line, 2976
Add separator line, 2976
Adding
  Adding times with T_ADD, 1668
Additional field devices (PROFIBUS and PROFINET), 488, 517
Address
  Area pointer, 2408
  Mitsubishi, 2615
  Omron Host Link, 2670
Address multiplexing
  with absolute addresses, 2144

with symbolic addresses, 2144
Address packing, 480
Address range
Changing, 432
Address register, 737
Addresses
Assigning, 433
Interrupt with packed, 485
Pack, 480
Unpack, 481
Addressing, 540
Addressing tags indirectly, 2139, 2140
Allen-Bradley Ethernet IP, 2570
Changing, 432
General, 430
MPI, 2501
Multiplexing, 2139
Addressing operands, 724, 725, 726, 728, 729, 730, 731, 732, 733, 734, 735, 738, 739
Adjust, 2361
AdjustContrast, 2298
Admin, 2828
Alarm
Acknowledge, 2835
Alarm data type, 1185
Components, 1182, 2167
Configuring, 2174, 2179
Displaying, 2098
Editing, 2836
Exporting, 2698
Formal parameter, 1185
Importing, 2700
in Runtime, 2829
In runtime, 2190
Inspector window, 48
System function, 2195
Alarm blocks, 1182
Alarm buffer
in Runtime, 2830
In runtime, 2191
Alarm buffer overflow, 2368
Alarm class, 2167
Copying, 1199
Creating, 1198
Editing, 1199
in Runtime, 2829
In runtime, 2191
Layout, 2192, 2832
Alarm classes, 2163
Custom, 2163
Name change through migration, 127
Predefined, 2163
Use, 2163
Alarm configuration
Additional text, 1192
Deleting an associate value, 1196
Entering text, 1191
Info text, 1191
Introduction, 1181
Locking an attribute, 1193
Locking text, 1192
Unlocking an attribute, 1193
Unlocking text, 1192
Alarm display, 641
"Active alarms" view,
Acknowledging an alarm, 645
Archive view, 642
Clear archive, 643
Export archive, 643
Ignoring alarms, 646
Layout of the alarms in the "Active alarms" view,
Layout of the alarms in the archive view, 642
Receiving alarms, 643
Using the keyboard, 646
Alarm editor
Structure, 1186
Alarm event
Acknowledge, 2161
Incoming, 2161
Outgoing, 2161
Alarm group, 2166, 2168
Configuring, 2171
Creating, 2171
Migration, 126
Alarm indicator, 47, 2102, 2183, 2194, 2834
Alarm classes, 2103
Application, 2194
Configuring, 2186
Events, 2103
in Runtime, 2831
In runtime, 2191
Layout, 2103, 2194
Operation, 2194
Alarm log
in Runtime, 2831
Alarm message
Acknowledgment by the PLC, 2458, 2518, 2558, 2602, 2629, 2656, 2674
Acknowledgment on the HMI device, 2459, 2518, 2559, 2602, 2630, 2657, 2675
Configure acknowledgment, 2458, 2517, 2558, 2601, 2629, 2656, 2674
Alarm number, 2166, 2167
Assigning, 1181

Alarm status, 2167
Acknowledged, 2161
Incoming, 2161
Outgoing, 2161
Alarm system, 2158
Alarm text, 2167
Formatting, 2177
Output fields, 51
Removing format settings, 2177
Special characters, 51
Alarm type, 1183
Editing, 1189
Alarm types, 2159
Alarm view, 47, 2098, 2100, 2182, 2193, 2833
Alarm text window, 2193, 2833
Column, 2099
Configuring, 2184
Configuring the layout, 2184
Control element, 2099
Layout, 2098, 2192, 2832
Operation, 2193, 2833
Alarm window, 47, 2182
Configuring, 2186
in Runtime, 2831
In runtime, 2191
Alarm with simple acknowledgment, 2165
Alarm without acknowledgment, 2165
Alarms
Output of a tag value, 2176
AlarmViewAcknowledgeAlarm, 2324
AlarmViewEditAlarm, 2323
AlarmViewShowOperatorNotes, 2325
AlarmViewUpdate, 2324
Align
Object flush, 1995
Allen-Bradley, 2561, 2584, 2586
Allen-Bradley DF1 communication driver, 2584, 2586
Analog alarm, 2601
Area pointer, 2599
Basic Panel, 2413
Communication drivers, 2562
Data type, 2601
DF1, 2562
EtherNet/IP, 2562
Mitsubishi, 2628
Allen-Bradley DF1
Configuring a connection, 2579
Connection, 2579, 2584
Connection parameters, 2582
CPU type, 2595
KF2 module, 2584, 2586
KF3 module, 2584, 2586
Migrating data types, 140
Valid data type, 2594
Allen-Bradley Ethernet IP
Address multiplexing, 2573
Addressing, 2570
Addressing type, 2572
Migrating data types, 141
Allen-Bradley EtherNet/IP
Configuring a connection, 2563
Connection, 2563, 2566
Connection parameters, 2564
Analog alarm, 2159
Allen-Bradley, 2601
Configuring, 2174, 2180
Limit value exceeded, 2133
Mitsubishi, 2628
Omron, 2673
Analog alarm types, 2159
Analog alarms
Configuring, 2168
AND, 1369, 1393, 1394, 1541
Logic operation, 1211, 1212, 1217
Animation
Configuring, 2040
Diagonal movement, 2044
Direct movement, 2044
Green arrow in Overview, 2040
Horizontal movement, 2043
Multiple selection, 2048
Object group, 2047
Overview, 2040
Vertical movement, 2044
ANY, 771
Appearance
Dynamization of an object, 2041
Application, 519
Alarm indicator, 2194
Button, 2817
Date/time field, 2809
Graphic IO field, 2813
Graphic view, 2813
IO field, 2810
Recipe view, 2105
Simple alarm view, 2192
simple alarm window, 2192
Switch, 2818
Symbolic IO field, 2820
Trend view, 2815
User view, 2085
Arccosine, 1317, 1489, 1604
Arcsine, 1316, 1487, 1603

Arctangent, 1605
Arctangent value, 1319, 1490
Area of unplugged modules, 332
Area pointer, 2441, 2502, 2542, 2676
  Acquisition cycle, 2408
  Active, 2408
  Address, 2408
  Allen-Bradley, 2599
  Basic Panel, 2416
  Comment, 2408
  Configuring, 2410
  Connection, 2407, 2411
  Connections editor, 2408
  Coordination, 2444, 2505, 2545, 2680
  Creating, 2410
  Data record, 2225, 2450, 2509, 2549, 2684
  Date/time, 52, 2441, 2443, 2543, 2678
  Date/time PLC, 52, 2504, 2544, 2679
  Job mailbox, 2448, 2507, 2547, 2682
  Length, 2408, 2410
  PLC tag, 2408
  Pointer name, 2408
  Project ID, 2447, 2506, 2546, 2681
  Screen number, 2446, 2502, 2542, 2677
  Tab, 2408
Area pointers, 2406
  Date/time, 2503
Arrange
  Object in the screen, 1986
Arrangement of byte sequence, 1511, 1615
Array, 50, 2150, 2152
  Creating, 2152
  Indirect addressing, 2139, 2150
  see ARRAY, 884, 1020
ARRAY
  Addressing, 729
  Declaration in global data blocks, 1020
  Declaration in PLC data types, 1040
  Declaration in the block interface, 884
  Example, 766, 767
  Format, 765
  Indirect addressing of components, 734
Array element
  Location of use of HMI tag, 50
  Name, 50
Array elements, 2150
ARRAY limits, 982
Array tag, 2150
  Char, 50
AS interface, 490
ASCII code table, 411
ASCII TSAP, 411
Asian characters
  Input on the HMI device, 2798
  Interpretation, 2799
  Memory requirements, 2798
Asian languages
  Configuration, 2799
  Font size, 2799
  Text field length, 2799
Asian operating system, 2725
ASIN, 1316, 1487, 1603
Assign
  Object of a layer, 2065
Assign symbol, 433
Assign tag, 433
Assigning
  a function to a function key, 2057
  a graphic to a function key, 2060
  Function key, 2051, 2053, 2055
Assigning an IP address, 38
Assigning parameters
  Hardware, 325
Assignment, 981, 1214, 1217, 1229, 1399
  Negate, 1230, 1400
Assignment list
  Enabling the display of retentive bit memories, 1102
Assignment list
  Defining filter,
  Delete filter,
  Displaying,
  Example for displaying bit memory,
  Example for displaying inputs and outputs,
  Filter options,
  Filtering,
  Introduction,
  Meaning of symbols,
  Setting view options,
  Structure,
Asynchronous
  Transferring data, 2451, 2510, 2555, 2685
Asynchronous error event
  Delaying with DIS_AIRT, 1744
  Enabling with EN_AIRT, 1745
AT, 40
AT, 40
ATAN, 1319, 1490, 1605
Authorization, 2821
  Assignment, 2260, 2281
  Changing the name, 2265
  configuring, 2273
  Creating, 2258, 2278
  Deleting, 2265
  Managing, 2263

Autocompletion
- Function, 871
- Insert tag, 872, 873
- Inserting an instruction, 873

Automation system, 2382, 2387
- Local, 3129
- Remote, 3129
- Setting up, 2382

Autonegotiation, 512, 2429, 2482
Availability
- Object for Basic Panels, 2083

Availability for specific devices
- Screen, 1969

AWP command, 444
AWP_In_Variable, 447
AWP_Out_Variable, 445
Axis technology object:Active homing, 2967
Axis technology object:Add new object, 2946
Axis technology object:Axis name configuration, 2948
Axis technology object:Basic parameters, 2947
Axis technology object:Changing the configuration parameters for dynamics in the user program, 2961
Axis technology object:Changing the configuration parameters for homing in the user program, 2968
Axis technology object:Commissioning overview, 2945
Axis technology object:Config.DriveInterface. tag, 3038
Axis technology object:Config.DynamicDefaults tag, 3041
Axis technology object:Config.DynamicLimits tag, 3040
Axis technology object:Config.General. tag, 3037
Axis technology object:Config.Homing tag, 3045
Axis technology object:Config.Mechanics tag, 3039
Axis technology object:Config.PositionLimits_HW tag, 3043
Axis technology object:Config.PositionLimits_SW tag, 3042
Axis technology object:Configuration overview, 2945
Axis technology object:Configuration window icons, 2947
Axis technology object:Configuring acceleration, 2956
Axis technology object:Configuring active homing, 2964
Axis technology object:Configuring approach/homing direction, 2964
Axis technology object:Configuring deceleration, 2956
Axis technology object:Configuring distance per motor revolution, 2950
Axis technology object:Configuring home position offset, 2965
Axis technology object:Configuring invert direction signal, 2951
Axis technology object:Configuring jerk limiter, 2957
Axis technology object:Configuring maximum velocity / start/stop velocity, 2956
Axis technology object:Configuring passive homing, 2963
Axis technology object:Configuring permit auto reverse at the hardware limit switch, 2964
Axis technology object:Configuring pulses per motor revolution, 2950
Axis technology object:Configuring ramp-down time, 2956
Axis technology object:Configuring ramp-up time, 2956
Axis technology object:Configuring reference point position, 2963, 2965
Axis technology object:Configuring smoothing time, 2957
Axis technology object:Configuring the homing speed, 2965
Axis technology object:Configuring the reference point switch input, 2963
Axis technology object:Configuring the side of the reference point switch, 2963, 2964
Axis technology object:Configuring the velocity limiting unit configuration, 2956
Axis technology object:Diagnosis overview, 2945
Axis technology object:Drive enable configuration, 2950
Axis technology object:Drive ready configuration, 2950
Axis technology object:Drive signal configuration, 2950
Axis technology object:Emergency stop deceleration configuration, 2959
Axis technology object:ErrorBits tag, 3051
Axis technology object:Extended parameters, 2947
Axis technology object:General dynamics configuration, 2955
Axis technology object:Hardware and software components, 2941
Axis technology object:Hardware interface configuration, 2948
Axis technology object:Mechanics configuration, 2950
Axis technology object:MotionStatus tag, 3047
Axis technology object:Passive homing, 2965
Axis technology object:PTO and HSC configuration, 2948
Axis technology object:Response when jerk limiter is activated, 2960
Axis technology object:StatusBits tag, 3049
Axis technology object:Tools, 2944
Axis technology object:Updating the tags, 3053
Axis technology object:User unit configuration, 2949

B

Background color
  Dynamization, 2041
Backing up, 2822
  HMI device data, 2794
Backup
  Data of the HMI device, 2778, 2779
Bar, 2084, 2808
  Color transition, 2084
  Display limit lines, 2084
Bar segment
  Defining, 2085
Basic mode, 1134
Basic Panel
  Area pointer, 2416
  Communication drivers, 2413
  Display and operating element, 2083
  ETHERNET, 2415
  HTTP protocol, 2413
  IF1B, 2415
  Interface, 2415
  Mitsubishi, 2413
  Modicon Modbus, 2413
  Omron, 2413
  OPC, 2413
  Runtime start, 2771, 2792
  S7 1200, 2413
  S7 200, 2413
  S7 300, 2413
  S7 400, 2413
Basics
  Migration, 115
Behavior
  Button, 2818
  Date/time field, 2809
  Graphic IO field, 2814
  IO field, 2811
  Simple recipe view, 2243, 2841
  Switch, 2819
  Symbolic IO field, 2820
Benefits of using TeleService, 3094
Bit (0, 1)
  Graphics list, 2034
  Text list, 2026
Bit field
  Resetting, 1234, 1404
  Setting, 1233, 1403
Bit logic instruction
  AND logic operation, 1211, 1212, 1217
  Assignment, 1214, 1217
  Normally closed contact, 1204, 1206, 1209
  Normally open contact, 1204, 1206, 1209
  OR logic operation, 1211
  Output coil, 1206, 1209
  Query positive signal edge of an operand, 1204, 1212
  Reset output, 1204, 1211, 1212
  Set output, 1204, 1211, 1212
Bit logic operations
  AND, 1393, 1394
  EXCLUSIVE OR, 1397
  Insert input, 1398
  OR, 1394, 1395
Bit memories
  Enabling the display of retentive bit memories, 1102
Bit number (0 - 31)
  Graphics list, 2036
  Text list, 2028
Bit strings, 746, 747, 748
Block
  Changing passwords for know-how protected blocks, 1091
  Closing, 860
  Comparing, 1056, 1057
  Comparing code blocks, 1054
  Comparing data blocks, 1055
  Comparison, 1052
  Compiling, 1071
  Compiling in the program editor, 1074
  Compiling in the project tree, 1073
  Consistency check, 1071, 1109, 1115, 1116
  Consistency check in the call structure, 1110
  Copying, 849, 850
  Deleting offline, 861
  Deleting online, 862
  Displaying properties, 858
  Downloading blocks to a memory card, 1084
  Downloading blocks to an accessible node, 1081
  Downloading to device, 1076, 1077
  Editing properties, 858
  Entering a comment, 852
  Entering a title, 851
  exporting to an external source file, 1049
  Fill, 1610
  Fill uninterruptible, 1336, 1508, 1612
  Filling, 1334, 1506
  Inserting, 849
  Know-how protection, 1085
  Leave, 1644
  Move, 1607
  Move operation, 1501
  Move uninterruptible, 1332, 1503, 1608
  Moving, 1330
  Opening, 859

Opening know-how protected blocks, 1089
Optimized access, 700
Pasting, 851
Printing know-how protected blocks, 1090
Properties, 854
Removing copy protection, 1087
Renaming, 861
Saving, 860
Setting up copy protection, 1087
Time stamp, 856
Types, 696
Uploading blocks from a memory card, 1085
Uploading blocks from an accessible device, 1083
Uploading blocks from device, 1082
Uploading from device, 1076
Using a library, 848
BLOCK, 774
Block access
Data block, 1025
Block call
Basics, 702
Calling as single instance or multi-instance, 704
Correcting the call type, 919, 957
Inserting, 917, 955, 995, 997, 998, 999, 1000, 1001
Multi-instance, 706
Nesting depth, 703
Single instance, 704
Updating, 919, 957, 1003
Block comment
Hiding, 870
Showing, 870
Block comparison
Basics, 1052
Carrying out offline/offline comparisons, 1057
Changing the block, 1070
Comparing code blocks, 1054
Comparing data blocks, 1055
Comparing online-offline, 1056
Comparison editor, 1059
Detailed comparison, 1066
Execute action, 1065
Filtering the comparison editor view, 1062
Navigation, 1069
Representation of the detailed comparison, 1067
Specifying actions, 1064
Synchronize scrolling, 1070
Updating comparison results, 1063, 1071
Block consistency
Checking, 1110
Checking in the dependency structure, 1116
Block folder, 844
Block interface
Declaring ARRAY, 884
Declaring PLC data type, 886
Declaring STRUCT, 885
Declaring tags, 881, 882, 887
Hiding, 870
Importing and exporting tags, 895
Multi-instance, 888
Purpose of tag declaration, 875
Retentivity, 891
Showing, 870
Structure, 876
Tag properties, 890, 891
Updating, 888
Valid data types, 879
Block parameters, 707, 708, 709, 710, 711, 712, 713, 714, 715
Basics, 581
Block interface, 876
Block property
Displaying, 858
Editing, 858
Function, 853
Overview, 854
BLOCK_DB_TO_WORD, 819
Bookmarks
Deleting, 990
Function, 988
Navigating, 989
Setting, 988
BOOL, 745, 783, 799
Borders
Insert background image, 248
Placeholder for document information, 251
Specifying the print area, 249
Boundaries, 514, 2431, 2483
Boundary reached
Event, 2369
Brackets, 980
Branch, 1631, 1634
Closing, 934
Definition, 932, 970
Deleting, 935, 971
Inserting, 933, 971
Rules, 933
Rules, 933
Button, 2107, 2817
Application, 2817
Behavior, 2818
Configuring, 2279
Configuring access protection, 2275
Define hotkey, 2108
Graphic, 2108

Layout, 2817
Mode, 2107
Operation, 2817
Text, 2108
Buttons and Switches
Library, 2073
BYTE, 746, 784, 800
Bytes
Swap, 1339, 1511, 1615

C

Cabling rules (PROFINET), 512, 2429, 2482
CALC, 1286, 1455
Calculate, 920, 958, 1286, 1455
CALCULATE, 920, 958
CalibrateTouchScreen, 2312
Call hierarchy, 703
Call structure, 1103
Displaying, 1107
Introduction, 1103
Meaning of symbols, 1105
Setting view options, 1108
Structure, 1106
Callback to a number specified during connection establishment, 3118
Canceling printing, 256
Cascaded counting function, 599
CASE, 1634
CEIL, 1344, 1516
Change word order
Modicon MODBUS TCP/IP, 2635
ChangeConnection, 2350
Changeover contact, 612
Changeover contact sensor type, 611
Changing
Displayed name of user group, 2264
Logoff time in runtime, 2270
Name of the user, 2270
Object size, 1992
Password, 2263
Password in runtime, 2270
User group in runtime, 2271
Users, 2827
Changing and displaying operating mode (example), 2291
Changing the name
Authorization, 2265
User group, 2264
Users, 2262
Char
Array tag, 50
CHAR, 763, 797, 816
Character, 763
Character string, 763
Combining character strings with CONCAT, 1693
Converting from a hexadecimal number with HTA, 1690
Converting number to character string with VAL_STRG, 1683
Converting numeric character string with STRG_VAL, 1681
Converting to a hexadecimal number with ATH, 1687
Converting with S_CONV, 1679
Copy character string to characters with Strg_TO_Chars, 1685
Copying characters to character string with Chars_TO_Strg, 1686
Deleting characters with DELETE, 1696
Determining length with LEN, 1692
Finding a character with FIND, 1699
Inserting characters with INSERT, 1697
Moving/copying with S_CONV, 1678
Reading out left character with LEFT, 1693
Reading out middle character with MID, 1695
Reading out right character with RIGHT, 1694
Replacing character with REPLACE, 1698
Check whether project can be migrated, 99
Checklist for troubleshooting the modem, 3121
Circle, 2094
Radius, 2094
CJ1, 2669
CJ2, 2669
Cleaning project for migration, 100
ClearAlarmBuffer, 2321
ClearAlarmBufferProTool, 2322
ClearAlarmBufferProtoolLegacy, 2322
cleared, 2360
Click, 2366
Clock memory, 548, 571
Closing
Runtime, 2793
CMP <, 1279, 1448
CMP <=, 1275, 1445
CMP <>, 1271, 1441
CMP ==, 1269, 1440
CMP >, 1277, 1447
CMP >=, 1273, 1443
Coil, 1229
Negate, 1230
Output coil, 1206
Color
Diagnostics of Ethernet cables, 653

Diagnostics of ports, 653
Color transition
Bar, 2084
Column
Alarm view, 2099
Comfort Panel
S7-1200, 56
Comment
Area pointer, 2408
Comments
Inserting in SCL program, 1003
Commissioning
HMI device, 2788
communication
Via the point-to-point port with MB_SLAVE, 40
Communication, 2381
Basics, 2382
Definition, 2381
S7 1200, 2417
S7 200, 2532
S7 300, 2470
S7 400, 2470
Third-party drivers, 2561
using area pointers, 2386
Communication drivers, 2386
Allen-Bradley, 2562
Basic Panel, 2413
Mitsubishi, 2603
Modicon Modbus, 2631
Omron, 2657
Communication network, 2387
PROFINET, 2388
Communication partners
HMI device, 2381
Networking, 2392
PLC, 2381
SIMATIC S7, 2469, 2529, 2538, 2541
Communications
Cycle load, 553
Communications instruction
"AS_DIAL",
"AS_MAIL",
"PG_DIAL",
"SMS_SEND",
Communications load, 548, 553
Communications module (CM), 1916
Communications modules, 561
Properties, 561
Communications port
Configuring, 562
Communications processor
Hotfixes, 35
Operation, 35
Communications protocol
Defining, 565
Overview, 40, 565
Compact, 2641, 2652
CompactLogix, 2569
Comparator
Greater than or equal to, 1214
Less than or equal to, 1214
Compare
Equal, 1269, 1440
Greater or equal, 1273, 1443
Greater than, 1277, 1447
Less or equal, 1275, 1445
Less than, 1279, 1448
Not equal, 1271, 1441
Compare offline/online
Automatic device assignment, 426
Compile
Address parameters, 56
Compiling
Block, 1073
Blocks, 1071
Consistency check, 1071
Correcting compilation errors, 1075
Migrated project, 119
Project, 2767
With disabled pulse generator, 39
Components
Alarm, 2167
Configuration
Downloading to device, 523
DP slave, 475
DP slave, simple, 467
Hardware, 325
Tag, 2128
Uploading to programming device/PC, 524
Configuration phase, 2787
Configure a PROFIBUS subnet
Matching parameters to one another, 359
Setting bus parameters, 360
Configure acknowledgment
Alarm message, 2458, 2517, 2558, 2601, 2629, 2656, 2674
Configured
Loop-in alarm, 2178
configuring
Authorization, 2273
Configuring, 2173, 2182
Access protection, 2279
Alarm acknowledgment, 2187, 2188, 2189
Alarm group, 2171

Alarm view, 2184
Analog alarm, 2174, 2180
Analog alarms, 2168
Connections, 2520, 2521, 2524, 2532
Controller alarms, 2168
Discrete alarm, 2172, 2179
Discrete alarms, 2168
Event-driven tasks, 2176
I slave, 477
Layout of the alarm view, 2184
Multiple tags, 2129
Recipe view, 2240
Rectangle, 2020
System events, 2168
Trend view for values from the PLC, 2155
Configuring a connection
Allen-Bradley DF1, 2579
Allen-Bradley EtherNet/IP, 2563
Mitsubishi FX, 2617
Mitsubishi MC TCP/IP, 2604
Modicon Modbus RTU, 2643
Modicon Modbus TCP, 2631
Omron Host Link, 2658
Configuring a PROFIBUS subnet
Meaning of the bus parameters, 361
Configuring the network with Ethernet, 365, 2426, 2479
Creating private subnets, 367, 2427, 2480
Linking networks, 368, 2428, 2481
Relationship between IP address and subnet mask, 367, 2427, 2480
Setting the IP address, 366, 2426, 2479
Setting the subnet mask, 366, 2426, 2479
Configuring the remote modem, 3098
Connecting
PLC, 2566, 2608, 2621, 2636, 2662
Connecting a TS Adapter with an external modem, 3099
Connecting a TS Adapter with an internal modem, 3098
Connecting cable
6XV1440 - 2P for Mitsubishi PG protocol, 2622
Allen-Bradley cable 1784-CP10, 2591
Mitsubishi FX, 2621
Connecting cables
Point-to-point cable 2, 2650
Connection, 369, 382, 2385
Address details, 690
Allen-Bradley DF1, 2579, 2584
Allen-Bradley EtherNet/IP, 2563
Area pointer, 2407, 2411
Configuring, 2393, 2397, 2563, 2579, 2604, 2617, 2631, 2643, 2658
Configuring a connection when there is no or no clear network assignment, 375
Creating, 398, 2393, 2397
Deleting, 400
Integrated, 2393
integrated connection, 2118, 2403
Mitsubishi MC TCP/IP, 2604
Modicon, 2647
Modicon Modbus RTU, 2643
Modicon Modbus TCP, 2631
offline, 2447, 2506, 2546
Omron Host Link, 2658
Parameters, 2564, 2582, 2606, 2618, 2633, 2645, 2660
S7 200, 2532
Table, 2391
Connection cable
Allen-Bradley cable 1747-CP3, 2592
Allen-Bradley cable 1761-CBL-PM02, 2593
Modicon, 2647
Multipoint cable 1, 2664
Multipoint cable 2, 2665
Omron Host Link, 2662
Point-to-point cable PP2 for Omron, 2666, 2667
Connection description
Changing parameter values, 407
Data block, 403
Structure, 404
Connection details, 689
Connection diagnostics
Detailed, 688
Overview, 686
Connection information, 688
Connection parameter assignment
Connection parameters, 395
General, 392
Overview, 393
Starting, 397
Connection parameters
Allen-Bradley DF1, 2582
Allen-Bradley EtherNet/IP, 2564
Mitsubishi FX, 2618
Mitsubishi MC TCP/IP, 2606
Modicon Modbus RTU, 2645
Modicon Modbus TCP, 2633
Omron Host Link, 2660
Connection resource, 370, 382
SIMATIC S7 1200, 54
Connection resources, 689
Connection status
Displaying using icons, 687
Connections

Allen-Bradley DF1, 2584, 2586
Configuring, 2520, 2521, 2524, 2532
Editor, 2520, 2521, 2524
Consistency
Slot rules, 327
Consistency check, 1109, 1110, 1115, 1116
Introduction, 1109, 1115
Constant
Basics, 718
Declaring, 835
Entering, 739
PLC tag table, 825
Properties, 839
Rules, 835
Context filter, 328
CONTINUE, 1641
Control element
Alarm view, 2099, 2101
Recipe view, 2105
simple alarm view, 2833
Simple alarm view, 2193
Trend view, 2095
Controller alarms
Configuring, 2168
ControlLogix, 2569
Conversion, 780
Explicit, 799, 800, 803, 804, 805, 807, 808, 809, 811, 812, 813, 814, 815, 816, 817, 819
Explicit, 799, 800, 803, 804, 805, 807, 808, 809, 811, 812, 813, 814, 815, 816, 817, 819
Explicit, 799, 800, 803, 804, 805, 807, 808, 809, 811, 812, 813, 814, 815, 816, 817, 819
Implicit, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798
Conversion operations
Converting times with T_CONV, 1668
Convert, 1341, 1513, 1616, 1617
CONVERT, 1341, 1513, 1616
Converted name (PROFINET), 499
Converting
A character string to a hexadecimal number with ATH, 1687
Character strings with S_CONV, 1679
Converting number to numeric character string with VAL_STRG, 1683
Converting numeric character string to number with STRG_VAL, 1681
Converting times with T_CONV, 1668
Copy numerical character string to characters with Strg_TO_Chars, 1685
Copying characters to numerical character string with Chars_TO_Strg, 1686
Hexadecimal number to character string with HTA, 1690
Converting an unspecified CPU, 153
Cookie, 448
Coordinated transfer, 2225
With PLC, 2225
Coordination, 2386
Copy
Block, 1330, 1607
Block uninterruptible, 1332, 1608
Screen, 1975
Tag, 2127
Template, 1981
Value, 1323
Copy protection, 1087
Copy template
Library, 2072
Copying
Adjusting screen size, 2761
Alarm indicator, 47
Alarm view, 47
Alarm window, 47
Color, 2761
Font, 2761
Function key, 2761
Hardware component, 342
HMI device, 43
Invalid object, 2760
Linked objects, 2763, 2764
Principle, 2759
Screen, 2761
User-defined folders, 2760
COS, 1313, 1484, 1600
Cosine, 1313, 1484, 1600
Count
Down, 1264, 1435, 1580
Up, 1262, 1433
Up and down, 1266, 1437, 1583
Count (8 DI NAMUR), 596
Counter
Count up and down, 1206, 1214
COUNTER, 774
Counter input, 557
Counter mode, 557
Counter operations
Control high-speed counters, 1848
High-speed counters, 1848
Counter, high-speed, 556
Counting
Up, 1577
Up and Down, 1206, 1214
Cover page

Insert background image, 248
Placeholder for document information, 251
CP1, 2669
CPM, 2669
CPU
Displaying the current LED status, 662
Fill level of all types of memory, 663
Inserting a signal board, 434
Properties, 547
Reading date and time-of-day with RD_SYS_T, 1673
Reading out a diagnostics buffer, 665
Selecting from the hardware catalog, 334
Setting time-of-day with WR_SYS_T, 1672
Switching operating mode, 670
CPU control panel
Display area, 662
CPU memory area
Displaying, 1120
CPU properties, 548
CPU type
Allen-Bradley DF1, 2595
Allen-Bradley EtherNet/IP, 2569
FX3 series, 2614
Mitsubishi FX, 2624
Mitsubishi MC TCP/IP, 2614
Modicon Modbus RTU, 2652
Modicon Modbus TCP, 2641
Omron Host Link, 2669
Create
Connection, 2419
Global library, 2080
Group, 2010
Template, 1980
Creating, 2232
Alarm group, 2171
Array, 2152
Connection, 2393, 2397, 2472
External tag, 2122
Infotext, 2173, 2175
Internal tag, 2124
Recipe, 2232
Recipe data record, 2843
Recipe data record on the HMI device, 2245, 2246
Screen, 1974
User group, 2259, 2281
Users, 2261, 2282, 2824, 2826
Users in runtime, 2268
Creating a print preview, 251
Creating a watch table, 1137
Crossing
Definition, 935
Deleting, 937
Inserting, 936
rearranging, 936
Cross-reference
Inspector window, 47
Cross-reference list, 1122, 2720
Displaying, 1124, 2720
Overview, 1124, 2720
Settings, 1122, 2720
Sorting columns, 1122, 2720
Structure, 1122, 2720
Views, 1122, 2720
Cross-references
Displaying, 1124, 2720, 2722
Introduction, 1121, 2719
Uses, 1121, 2719
CS1, 2669
CSV file, 2694, 2696, 2706
CTD, 1264, 1435, 1580
CTRL_HSC, 1848
CTU, 1262, 1433, 1577
CTUD, 1206, 1214, 1266, 1437, 1583
Custom alarm classes, 2163
Custom web pages, 439, 455
Cycle load, 553
Cycle monitoring time, 1646
Cycle time, 548, 552
display configured, 658
display measured, 661
Cyclic
Continuous, 2136
In operation, 2136
Cyclic interrupt OB
Assigning parameters, 583
Description, 577
Querying parameters with QRY_CINT, 1733
Setting parameters with SET_CINT, 1731
Cyclic OB
Description, 574
Cyclic operation, 2531, 2534
Cyclic program execution
Options for interrupting, 574
Programming, 574

D

Data areas
Area pointer, 2409, 2677
Reading, 2409, 2677
Writing, 2409, 2677
Data backup, 69
HMI device, 2778, 2794

Data bit (DBX), 535
Data block
  Based on a PLC data type, 1022
  Creating, 846, 1017
  Declaration table, 1015
  Declaring ARRAY, 1020
  Declaring STRUCT, 1021
  Declaring tags, 1022
  Define start values, 1023, 1024
  Displaying values, 1036
  Global data block, 698, 1014
  Importing and exporting tags, 1034
  Instance data block, 699, 1014
  Monitoring data values online, 1034, 1035
  Optimized access, 700
  Programming, 1014, 1019
  Reading from load memory with READ_DBL, 1767
  Retentive behavior, 1025, 1026
  Tag properties, 1028, 1029, 1030
  Updating, 1018
  Writing to load memory with WRIT_DBL, 1769
Data byte (DBB), 535
Data consistency, 1850
Data double word (DBD), 535
Data exchange, 2381
  DP slave, 467
  I slave - DP master, 468
  using area pointers, 2386
Data exchange over the AS-AS remote link
  Communication instruction "AS_DIAL",
Data flow, 2839
Data flow control, 563
Data log, 438
  Closing with DataLogClose, 1764
  Creating with DataLogCreate, 1758
  Creating with DataLogNewFile, 1765
  Creating with DataLogWrite, 1763
  Opening with DataLogOpen, 1761
Data mailbox
  For recipes, 2225
Data record, 2386
  Reading, 2247
  Reading with RDREC, 1700
  Transfer, 2247
  Transferring with WRREC, 1702
Data record list, 2229
Data type
  Allen-Bradley, 2601
  ANY, 771
  ARRAY, 765
  BOOL, 745, 783, 799
  BYTE, 746, 784, 800
  CHAR, 763, 797, 816
  Conversion, 780, 786, 794, 795, 796
  Convert explicit, 1513
  Convert explicitly, 1616, 1617
  DATE, 759, 796, 814
  DINT, 753, 791, 808
  Discrete alarm, 2601, 2628, 2673
  DT, 761
  DTL, 762, 795, 815
  DWORD, 748, 786, 802
  Explicit conversion, 799, 800, 801, 802, 803, 804, 805, 807, 808, 809, 811, 812, 813, 814, 815, 816, 817, 1341
  Implicit conversion, 783, 784, 785, 787, 788, 789, 790, 791, 792, 793, 794, 795, 797, 798
  INT, 751, 789
  Internal tag, 2121, 2406
  LREAL, 756, 794, 812
  Mitsubishi, 2628
  Mitsubishi FX, 2623
  Mitsubishi MC TCP/IP, 2567, 2613
  Modicon Modbus RTU, 2651
  Modicon Modbus TCP, 2639
  Omron, 2673
  Omron Host Link, 2668
  PLC data type, 775, 1036
  POINTER, 769
  REAL, 755, 793, 811
  S5TIME, 758
  SINT, 749, 787, 803
  STRING, 763, 798, 817
  STRUCT, 768
  TIME, 757, 794, 813
  TIME_OF_DAY, 760
  TOD, 795, 814
  UDINT, 754, 792, 809
  UINT, 752, 790, 807
  USINT, 750, 788, 804
  Valid, 2567, 2613, 2623, 2639, 2651, 2668
  Validity, 741
  VARIANT, 773
  WORD, 747, 785, 801
Data types
  Migration, 139
  S7 1200 V1, 2460, 2461
  S7 200, 2560
  S7 300/400, 2519
  Valid, 2460, 2461, 2519, 2560
Data word (DBW), 535
Date, 759, 761, 762
DATE, 759, 796, 814
Date/time, 2386

Area pointer, 2441
Date/time field, 2086
Application, 2809
Behavior, 2809
Display system time, 2086
Format, 2087
Layout, 2809
Operation, 2809
Using tags, 2087
Date/time PLC, 2386
DATE_AND_TIME, 761
DB variable
Definition, 717
Deactivate, 2364
Deadband
Setting, 2175
DEC, 1299, 1468
Decimal places, 32, 1320, 1492, 2804, 2806
DECO, 1375, 1548, 1651
Decode, 1375, 1548, 1651
DecreaseFocusedValue, 2348
DecreaseTag, 2349
Decrement, 1299, 1468
Default font, 2746, 2797
Defective devices, 648
Defective license, 70
Define
Reference object, 1993
Define hotkey, 2108
Defining
For the assignment list, 1100
Delay time
Setting, 2175
Delete
Template, 1981
Deleting
Authorization, 2265
Filter in the assignment list, 1100
Hardware component, 341
Object, 1989
Recipe data record, 2845
Screen, 1975
Tag, 2128
User group, 2264
Users, 2263, 2828
Deleting a connection, 158
Demultiplex, 1382, 1555, 1657
DEMUX, 1382, 1555, 1657
Dependency structure, 1111
Introduction, 1111
Meaning of symbols, 1113
Setting view options, 1114
Structure, 1112
Dependency structure
Displaying,
Designation
TS Adapter, 3101
Designing a background, 2001
Designing a frame, 2001
Details view, 189
Determine diagnostics status, 664
Determining time of day, 672
Device
Adding to a hardware configuration, 336
Copying, 43, 342
Defective devices, 648
Deleting, 341
Inserting, 2383
Moving, 344
Renaming, 425
Device address, 430
Device information, 240, 648
Device name, 493, 496, 499
Device number, 494
Device overview
Address range, 430
Device replacement without exchangeable medium, 514
Device Tool, 521, 522
Device view
Area of unplugged modules, 332
Edit parameters, 345
Edit properties, 345
Hardware and network editor, 310
Inserting a signal board, 434
Inserting module, 339
Racks, 329
Device wizard, 2692
Devices
Connecting, 2385
Networking, 2384, 2391, 2392
Devices & Networks, 2384, 2391
Devices and networks, 2384, 2391
Connections, 2393
HMI connections, 2393
Diagnose repeater, 683
Diagnostic data
Reading out with GET_DIAG, 1750
Diagnostic error interrupt OB, 580
Diagnostics, 607
DP slave, reading diagnostics data with DPNRM_DG, 1725
Diagnostics buffer
Basics, 539, 678

Organization, 678
Diagnostics status
Determine and display online, 649
Diagonal movement
Animation, 2044
DINT, 753, 791
DINT, 753, 791
Direct access to the I/O, 540
Direction output and travel direction:relation, 2934
Disabling
Project language, 2727
Discrete alarm, 2159, 2161
Allen-Bradley, 2601
Configuring, 2172, 2179
Mitsubishi, 2628
Omron, 2673
Discrete alarm types, 2159
Discrete alarms
Configuring, 2168
display
Tooltip, 2807
Display object
Availability for Basic Panels, 2083
Display PDF files, 61
Display system time, 2086
Display the memory types of a CPU, 663
Display Welcome Tour, 61
Displaying
Assignment list, 1097
Call structure, 1107
CPU memory area, 1120
Cross-reference, 2722
Cross-references, 1124, 2720
Dependency structure, 1113
Load memory, 1120
Maximum load memory available, 1120
Program information, 1093
Displaying a force table, 1161
Displaying alarms, 641
Displaying the online status, 3075
Displaying values
As a trend, 2155
Distribute
Objects evenly, 1995
Distributed I/O, 465, 605
DIV, 1294, 1463
Divide, 1294, 1463
Documentation editor
Structure, 245
Documentation function
Using frames and cover pages, 243
Double word, 748
double-click, 2364
Download
Error messages, 2782
Extended download to device, 56
Project, 2770, 2788
Downloading
Downloading blocks to device, 1077, 1079
to the device, 523
Downloading data to the PLC
Error message, 56
DP, 2440, 2491
DP interface, 470
DP master, 592
Add DP master system, 473
Devices and modules, 470
Disconnection from DP master system, 472
Display on DP slave, 471
DP interface, 470
DP master system
Add DP slave, 474
Creating, 466, 468, 470
Disconnect DP slave, 475
Disconnection from subnet, 472
Edit properties, 473
Highlight, 471
Node disconnection, 472
DP slave
Add to DP master system, 474
Assign DP master system, 475
configuring, 467, 475
Data exchange, 467
Disconnect from DP master system, 475
DP master display, 471
Hardware catalog, 466
Intelligent, 476
Networking, 468
Reading diagnostics data with DPNRM_DG, 1725
Types, 473
With preprocessing, 468
DP standard slave, 488
DP standard slaves
Reading consistent data with DPRD_DAT, 1721
Writing consistent data with DPWR_DAT, 1723
DP/DP coupler, 467
DPV1
Configuring an ET 200S, 484
Drive signals, 2932
DT, 761
DTL, 762, 795, 815
DTL data type
Restriction, 52
Duration, 758, 1431

DWORD, 748, 786, 802
Dynamization
    Color of an object, 2041
    Flashing, 2041
    Object, 2038
Dynamize
    Appearance of an object, 2041
    Direct movement, 2044
    Green arrow in Overview, 2040
    Movement of an object, 2043

E

Eastern characters
    Input on the HMI device, 2798
Edge
    Negative, 1239, 1241, 1244, 1409, 1412, 1414
    Positive, 1237, 1240, 1243, 1408, 1410, 1413
    Query positive signal edge of an operand, 1204, 1212
Edit
    Object within a group, 2014
Edit networking
    Disconnecting from a network, 355
EditAlarm, 2303
Editing
    Alarm, 2836
    Folder link, 2006
    Function list, 2290
    Recipe data record, 2844
Editing language, 2723
    Selecting, 2728
Editing networking
    Copying a subnet, 356
    Copying subnets and devices, 357
Editor
    Connection, 2397
    Connections, 2520, 2521, 2524
    Graphics, 2737
Element list, 2229
Ellipse
    Horizontal radius, 2090
    Vertical radius, 2090
Ellipse, 2089
ELSE, 1634
E-mail
    Transferring e-mail with TM_MAIL, 1958
Empty box
    Inserting a LAD element, 912
    Inserting an FBD element, 951
EN/ENO mechanism
    Basics, 819
    FBD example, 821
    LAD example, 821
    SCL example, 822
    STL example, 823
enable, 2361
Enable autonegation, 512, 2429, 2482
ENCO, 1377, 1549, 1653
Encode, 1377, 1549, 1653
End of discovery of accessible devices, 514, 2431, 2483
End of sync domain, 514, 2431, 2483
End of topology discovery, 514, 2431, 2483
Engineering system
    Performance features, 2848
Enumeration types, 450
Error alarm
    Acknowledge, 2835
Error analysis with RET_VAL, 1222
Error handling, 1013
    Basics, 1009
    GetError, 1363, 1535, 1646
    GetErrorID, 1367, 1539, 1650
    Local error handling, 1011
Error information, 1011, 1222, 1363, 1367, 1535, 1539, 1646, 1650
Error message
    Downloading data to the PLC, 56
Error messages
    Download, 2782
Establishing a connection from a remote system ("PG_DIAL"),
Establishing a remote connection, 3119
    Procedure, 3120
ET 200eco, 589
ET 200eco PN, 589, 605
ET 200iSP, 588
ET 200iSP distributed I/O station
    Definition, 591
ET 200L, 588
ET 200M, 588
ET 200pro, 589
ET 200R, 589
ET 200S, 588
    DPV1 mode, 484
    Option handling, 482
    Positioning module, 37
    Pulse generator module, 37
    Reference junctions, 479
    Slot rules, 478
ET 200S COMPACT, 588
Ethernet
    S7 200, 2466, 2526, 2536

ETHERNET
- Basic Panel, 2415

Ethernet interface
- Displaying parameters, 3080

Ethernet module
- Removal/insertion, 36

Event
- Acknowledge, 2365, 2368
- Adjust, 2361
- Alarm buffer overflow, 2368
- Assigning organization block (OB) with ATTACH, 1729
- Boundary reached, 2369
- Canceling assignment to organization block (OB) with DETACH, 1730
- cleared, 2360
- Click, 2366
- Deactivate, 2364
- double-click, 2364
- Enable, 2361
- Execute, 2362
- For function lists, 2141
- Free space critically low, 2371
- Incoming, 2366
- Inspector window, 48
- Limit, 2367, 2368
- Loop-In-Alarm, 2367
- Low free storage space, 2371
- Outgoing, 2366
- Overflow, 2370
- Press, 2365
- Press key, 2370
- release, 2367
- Runtime Stop, 2369
- Screen change, 2363
- Selection changed, 2362
- Switch off, 2371
- Switch on, 2371
- Tags, 2141
- Time expired, 2372
- User change, 2363
- Value change, 2372
- When dialog is closed, 2362
- When dialog is opened, 2362

Event trigger, 2377, 2380

Event-driven tasks
- Configuring, 2173

Example
- Application for alarm classes, 2163
- Changing and displaying operating mode, 2291
- For displaying bit memory in the assignment list, 1095
- For displaying inputs and outputs in the assignment list, 1095
- For entering force values in the force table, 1164
- Modify values in the watch table, 1141
- System alarm, 2160

Example:,

Examples
- Controlling a conveyor belt using bit logic instructions, 1203, 1211
- Controlling room temperature using timer and bit logic instructions, 1209, 1217
- Detecting the direction of a conveyor belt using bit logic instructions, 1204, 1212
- Detecting the fill level of a storage area using counter and comparison instructions, 1206, 1214

Exchangeable medium, 514

EXCLUSIVE OR, 1396, 1397
- Bit logic operations, 1396

Execute, 2362

Exit
- User program, 1362, 1534, 1645

EXIT, 1642

EXP, 1310, 1481, 1598

Expanded mode, 1134

Expanding and collapsing sections of code, 987

Exponential value, 1310, 1481

Exponentiate, 1322, 1493

Export
- Alarm, 2698
- Project texts, 2716, 2734
- Recipe, 2694
- Tag, 2706
- Text list, 2711

Expression
- Arithmetic expression, 976
- Basics, 975
- Logical expression, 979
- Operator precedence, 980
- Relational expressions, 978

EXPT, 1322, 1493

Extended download to device, 56

Extended status information, 982

External graphic
- Edit folder, 2005
- Link folder, 2006
- Removing the folder link, 2005
- Rename folder, 2005

External image file
- add to graphics library, 2739
- Managing, 1987
- Storing in the image browser, 2007

External source file

- Basics, 1047
- Editing, 1050
- Exporting blocks, 1049
- inserting, 1049
- Linking file types to an editor, 1050
- Opening, 1050
- Rules for programming, 1048

F

FAQs, 322
Favorites
- Adding, 915, 953, 993
- Hiding, 870
- Removing, 916, 954, 995
- Showing, 870
- Using, 915, 954, 994

FB, 697
FBD, 939
- Migration, 107

FBD element
- Copying, 963
- Cutting, 963
- Deleting, 967
- Inserting, 950, 951
- Inserting an operand, 968
- pasting from the clipboard, 964
- Replacing, 965
- Rules for inserting, 948
- Selecting, 962

FBD network
- Branch, 970
- Deleting a branch, 971
- Inserting a block call, 917, 955
- Inserting a branch, 971
- Program status display, 1129
- Rules for branches, 970

FC, 697
Field, 765
Field device, 488, 517
Fill
- Block, 1334, 1506, 1610
- Block uninterruptible, 1336, 1508, 1612

FILL_BLK, 1334, 1506, 1610
Filter
- Defining for the assignment list, 1100
- Delete, 1100
- Hardware catalog, 328
- In the assignment list, 1099
- Selecting, 1101

Filters
- Assignment list, 1101

Find and replace, 261
- Additional options for searching, 262
- Replacing search keys, 263
- Start search, 262
- Using the search function, 262

Firmware update memory card, 533
Flash test, 682
Flashing, 2041
Flip
- Object, 1986

Flip-flop
- Reset/set, 1236, 1406
- Set/reset, 1235, 1405

Floating-point number, 755, 756
- Check invalidity, 1285, 1454
- Check validity, 1284, 1453

FLOOR, 1345, 1517
Flow control, 38
Flutter monitoring, 612
Folder link
- Editing, 2006
- Removing, 2006
- Renaming, 2006

Font settings
- Migrate, 124

Font size
- Asian languages, 2799

FOR, 1636
Force
- Force all, 1176
- Stop forcing, 1178, 1179

Force job
- On SD card, 30

Force table
- "Monitor once and now" command for tags,
- Basic mode, 1160
- Display, 1161
- Expanded mode, 1160
- Functionality, 1156
- Layout, 1159
- Meaning of the columns, 1159
- Meaning of the icons, 1160
- Monitoring and modifying modes, 1145
- Opening, 1161
- Overview of the display formats, 1165
- Overview of the test options, 1156
- Permitted operands, 1163
- Permitted operands for force values, 1164
- Saving, 1162
- Switching between basic mode and expanded mode, 1160
- Syntax check, 1162

Test options, 1156
Force value
Permitted operands, 1164
Forcing tags
Safety precautions, 1158, 1173
Foreground color
Dynamization, 2041
Form exponential value, 1598
Format, 2087
Formatting
Alarm text, 2177
FRAC, 1320, 1492
Fragment, 452, 458
Free space critically low, 2371
Free-form comments
Deleting, 923, 961
Editing, 922, 960
Inserting, 922, 960
Introduction, 921, 959
Frequency meter, 602, 605
Function
Assigning to a function key, 2057
Function (FC)
Creating, 845
Definition, 697
exporting to an external source file, 1049
Function block (FB)
Creating, 845
Definition, 697
exporting to an external source file, 1049
Instance data block, 697
Function Block Diagram, 939
Function key, 2051
Assigning a function, 2057
Assigning a graphic, 2060
Global assignment, 2053
Global screen, 1979
Local assignment, 1979, 2055
protect with password, 2058
Use global assignment, 1979
used for screen navigation, 2062
Function list, 2141, 2287, 2376
Asynchronous execution, 2291
Execution, 2288
Execution in Runtime, 2291
Synchronous execution, 2291
System function, 2288
User-defined functions, 2288
Function lists
Editing, 2290
Functional scope
ProSave, 2777
Functionality of the force table, 1156
Functions
Updating tag value, 2118, 2403
FX1 series, 2624
FX2 series, 2624
FX3 series, 2614

G

Gateways, 3097
General information on troubleshooting, 3134
Generation of packed addresses, 481
GetError, 1012, 1013, 1363, 1535, 1646
GetError, 1012, 1013, 1363, 1535, 1646
GetErrorID, 1011, 1012, 1013, 1367, 1539, 1650
GetGroupNumber, 2319
GetPassword, 2319
GetUserName, 2318
Global assignment
Of a function key, 2051, 2053
Global data block, 698
Configuring, 2410
Creating, 2410
Retentive behavior, 1026
Global library, 2070
Adding a copy template, 293
Adding types, 294
Basics, 285
Closing, 290
Create, 2080
Creating, 286
Creating folders, 292
Deleting, 291
Editing elements, 298
Migrating, 132
opening, 2082
Opening, 287
Removing elements, 300
Save, 2081
Saving, 289
Shared work, 286
Showing properties, 288
Using a supplied global library, 301
Using filter view, 285, 302
Using master copies, 295
Using the element and part views, 276
Using types, 296
Global screen, 1978
Function key, 1979
Template, 1980
Global softkey, 2053
Global tag, 825

Global telephone book properties, 3096
Go online
    connecting several devices, 3076
Going offline, 681
Going online, 680
    multiple TIA portal instances, 36
GOTO, 1643
GoToEnd, 2308
GoToHome, 2308
GRAPH
    Migration, 108
Graphic
    add to graphics library, 2737
    Assigning to a function key, 2060
    Button, 2108
    Graphic view, 2091
    Inserting, 1987
    managing, 2005
    Stretching, 2091, 2093
    Using from the graphic browser, 2004
    With transparent background, 2004
Graphic browser, 2004
Graphic browser, 2004
Graphic I/O field, 2092
    Output graphics list, 2037
Graphic IO field, 2813
    Application, 2813
    Behavior, 2814
    Operation, 2814
Graphic view, 2090, 2813
    Application, 2813
    Layout, 2813
    Operation, 2813
    transparent color, 2091
Graphics, 2007
    Editor, 2737
Graphics list
    Application, 2029
    Bit (0, 1), 2034
    Bit number (0 - 31), 2036
    Creating, 2030
    Graphic I/O field, 2037
    Outputting configuration data, 2036
    Range (0 - 31), 2031
    Range (... - ...), 2033
Greater than or equal to, 1214
group
    Cancel, 2011
    Edit, 2010
    Ungroup, 2011
Group
    Adding objects, 2012
    Cancel, 2011
    Creating, 2010
    Editing, 2010
    Removing an object, 2013
    Ungroup, 2011
GSD files
    Configuring devices (PROFIBUS), 488
    GSD revisions (PROFIBUS), 486
    Installation, 487
GSD files (PROFINET), 515
    Changing revision, 517
    Installation, 517
GSDML, 515

H

Handshaking, 564
Hardware
    Configurimg and assign parameters, 325
    Detection, 38
    Edit parameters, 345
    Edit properties, 345
Hardware and network editor
    Device view, 310
    Network view, 307
    Topology view, 313
Hardware and software limit switches:Function, 2935
Hardware catalog
    Adding device, 337
    Browsing, 327
    DP slave, 466
    DP/DP coupler, 467
    I slave, 466
    Selecting the hardware component, 334
    Task card, 320
Hardware configuration
    Adding device, 336
    Adding module, 339
Hardware configuration for Motion Control S7-1200, 2929
Hardware diagnostics, 647
Hardware editor
    Components, 305
    Function, 305
    Hardware catalog, 320
    Inspector window, 318
Hardware ID, 430
Hardware interrupt OB
    Assigning parameters, 581
    Description, 578
Hardware requirements, 62
    Communication instruction "AS_MAIL",

Communication instruction "SMS_SEND",
Communications instruction "PG_DIAL",
Hardware requirements for AS-AS remote link
Communication instruction "AS_DIAL",
Hardware support package
Installing, 76
Hardware-controlled data flow control, 564
Harmonize
Object size, 1996
HART, 593
Help
Browsing, 204
Call topic from favorites, 205
Delete topic from favorites, 205
Find keywords, 203
Full-text search, 204
Identification of Help topics, 201
Open, 203
Open tooltip cascades automatically, 206
Printing help topics, 206
Save topics in favorites, 205
Use index, 203
Help indicator, 2093
Hibernate, 30
Hidden input, 2089
Hidden parameters, 931, 969
High-speed counter
Configuring, 559
General, 556
How it works, 556
HMI connection, 155, 382, 2385
Creating, 2419, 2472
MPI, 2493
MPI parameters, 2495
PROFIBUS, 2485
PROFIBUS parameters, 2434, 2487
PROFINET, 2418, 2419, 2471, 2472
PROFINET parameters, 2421, 2474
S7 1200, 2419
S7 300/400, 2472
HMI connections
Devices & Networks, 2393
HMI device, 67
Changing the device type, 2784
Commissioning, 2788
Data backup, 2778, 2779, 2794
MPI parameters, 2497
Performance features, 2850
PROFIBUS parameters, 2436, 2488
PROFINET parameters, 2422, 2475
Recommissioning, 2788
Reset to factory settings, 2782
Restoring data, 2778, 2779, 2794
System limits, 2850
HMI device licensing
Non-PC-based, 67
HMI device replacement, 44
HMI device type
Changing, 2784
HMI device wizard, 2692
Homing:Homing modes, 2937
Horizontal radius, 2090
Hotfix for communications processor, 35
How the frequency meter works, 603
HSC, 556
HSP, 76
HTTP protocol
Basic Panel, 2413
HTTPS, 438

I

I address, 430
I slave, 468, 476
configuring, 477
Data access, 477
Data exchange, 468
Hardware catalog, 466
I/O
Direct access to, 540
I/O access error, 538
System reaction, 37
I/O address, 430, 540
I/O field, 2087
Data format, 2088
Hidden input, 2089
Mode, 2088
I/O input, 535
I/O output, 535
I/Os, 725
Icon
For comparison, 652
For comparison status, 688
for connection status, 687
For diagnostics, 651
for operating mode, 653
Overlay icon, 652
Icons
in TeleService, 3124
Icons in the force table, 1160
Icons in the watch table, 1136
IE/AS-i link PN IO, 490
IEC Check, 780
Setting, 782

IEC time, 757
IEEE tag, 592
IF, 1631
IF1B
  Basic Panel, 2415
Image file
  Storing in the image browser, 2007
  Storing in the image browser, 2007
Import
  Alarm, 2700
  Analog alarm structure, 2701
  Project texts, 2718, 2736
  Recipe, 2696
  Structure of discrete alarms, 2704
  Structure of recipe data, 2697
  Tag, 2707
  Text list, 2713
Importing a recipe
  Structure for the import, 2697
Importing analog alarms
  Structure for the import, 2701
Importing discrete alarms
  Structure for the import, 2704
in the screen
  Arrange object, 1997
IN_RANGE, 1281, 1450
INC, 1298, 1467
Incoming, 2161, 2366
IncreaseFocusedValue, 2306
IncreaseTag, 2307
Increment, 1298, 1467
Indenting and outdenting
  Lines, 986
Index tag, 2139, 2140
Indirect addressing, 732, 733, 734, 735, 738
Indirect addressing, 732, 733, 734, 735, 738
Industrial Ethernet, 2387
Information system
  Components of the information system, 200
  Roll-out, 201
  Tooltip, 202
Infotext, 2168
  Creating, 2173, 2175
Initialization (custom web pages), 440
Initialization of the modem, 3099
Initialization string requirements, 3099
Input
  Inserting, 927, 1398
  remove, 928, 966
Input (I), 535
Input byte (IB), 535
Input word (IW), 535
Insert
  Library object, 2079
Insert empty line, 2976
Insert input, 965
Insert separator line, 2976
Inserting
  Graphic, 1987
  Object, 1986, 1987
  Rectangle, 2020
Inspector window
  Cross-reference, 47
  Event, 48
  Layout, 184
  Reducing automatically, 178
Inspector window
  Hardware and network editor,
Inspector window
  Hardware and network editor,
Inspector window
  Diagnostics tab,
Installation
  Displaying software, 77
  Licenses, 71
  Log, 72
  Modifying products, 78
  ProSave, 55
  Repairing products, 80
  Starting, 73
  Support package, 76
  System requirements, 61
  Target directory, 33
  Updating products, 78
Installing support packages, 76
Installing the local modem, 3098
Instance data block
  Creating, 699
  Definition, 699
  Retentive behavior, 1025
Instance DB
  Creating, 1189
Instruction
  Copying, 1006
  Cutting, 1006
  Deleting, 1006
  Inserting, 1006
  Rules, 991
  Versions, 897
Instructions for configuring a remote modem, 3099
INT, 751, 789, 805
Integer, 1347, 1519, 1626
  16-bit, 751, 752
  32-bit, 753, 754

8-bit, 749, 750
Integrated
Connection, 2393, 2397
Integrated connection, 2118, 2403
Integrated project
Converting an unspecified CPU, 153
Creating an integrated HMI connection, 155
Deleting an unspecified connection, 158
Linking HMI tags, 157
Migrating, 135, 150
Post-editing, 153
Integrating texts from text lists in alarms, 1197
Example, 1198
Intelligent DP slave, 476
Interconnecting ports
Graphic view, 423, 424
In the Inspector window, 508
Table view, 423, 425
Interface, 659
Adding, 3078
Basic Panel, 2415
Displaying, 659
Renaming, 425
Interface properties, 659
Interrupt event
Delaying with DIS_AIRT, 1744
Enabling with EN_AIRT, 1745
Interrupts
Activating time-of-day interrupt with ACT_TINT, 1737
Canceling time-delay interrupt with CAN_DINT, 1742
Canceling time-of-day interrupt with CAN_TINT, 1736
Querying time-delay interrupt with QRY_DINT, 1743
Querying time-of-day interrupt with QRY_TINT, 1738
Receiving from I/O module with RALRM, 1704
Setting time-of-day interrupt with SET_TINTL, 1735
Starting time-delay interrupt with SRT_DINT, 1741
With packed addresses, 485
INV, 1374, 1546
Invalid license
With a time zone change, 70
InverseLinearScaling, 2311
Invert, 1228, 1399
InvertBit, 2309
InvertBitInTag, 2310
Inverting, 1374, 1546
InvertLinearScaling, 2311
IO device
Networking, 506
Update time, 509
Watchdog time, 509
IO field, 2810
Application, 2810
Behavior, 2811
Changing between input fields, 2811
IO field, 2812
Layout, 2811
Operation, 2811
Touch operation, 2812
IO system, 505
Creating, 506
Send clock cycle, 508
IO-Link, 522
IP address, 31, 493
Assigning, 672
IP address parameters, 495
IP configuration
Change parameters from the user program, 1887
IP parameters, 659
IQ sense, 615
iQ series, 2614
ISO-on-TCP
Characteristics, 402
TSAP, 409

J

Jerk limiter:Function, 2936
JMP, 1352, 1524
JMP_LIST, 1355, 1527
JMPN, 1353, 1525
Job mailbox, 2386
Transferring data, 2453, 2512, 2551, 2688
Jump, 1352, 1353, 1354, 1355, 1357, 1360, 1524, 1525, 1526, 1527, 1529, 1532
Jump distributor, 1357, 1529
Jump label, 1354, 1526, 1643
Jump list, 1355, 1527

K

Key operation
Date/time field, 2810
IO field, 2812
Keyboard shortcuts
Editing texts, 196
Project editing, 194
Project tree, 195
Tables, 195
Window, 195

Keyword highlighting, 982
Keywords, 719
Know-how protection
  Changing a password, 1091
  Converting blocks from V10.5, 40
  Introduction, 1085
  Opening blocks, 1089
  Printing block, 1090
  remove, 1092
  Setting up, 1088

L

LABEL, 1354, 1526
LAD, 898
  Migration, 107
LAD element
  Copying, 925
  Cutting, 925
  Deleting, 929
  Inserting, 911, 912
  Inserting an operand, 930
  pasting from the clipboard, 926
  Replacing, 927
  Rules for inserting, 908
  Selecting, 924
LAD network
  Branch, 932
  Closing a branch, 934
  Crossing, 935
  Deleting a branch, 935
  Deleting a crossing, 937
  Insert crossing, 936
  Inserting a block call, 917, 955
  Inserting a branch, 933
  Program status display, 1128
  Prohibited interconnections, 910
  Rearranging a crossing, 936
  Rules for simultaneous branches, 933
Ladder Logic, 898
Language
  Activate project language, 2727
  Asian languages, 2725
  Asian operating system, 2725
  Disabling the project language, 2727
  Editing language, 2728
  Language support, 2726
  Language-dependent format, 2725
  Language-specific graphic, 2737
  Log, 2747
  multilingual project, 2729
  Reference language, 2728
  Regional format of the date, time, currency, and numbers, 2725
Language abbreviation, 462
Language behavior
  Screen keyboard, 58
Language switching, 462, 2741, 2798
  Log, 2747
  Runtime language, 2744
Languages
  Migrating, 129
Layer
  Assigning objects to a layer, 2065
Layout
  Alarm indicator, 2103, 2194
  Alarm view, 2098, 2101, 2192, 2832
  Bar, 2084
  Button, 2107, 2817
  Circle, 2094
  Date/time field, 2086, 2809
  Ellipse, 2089
  Graphic I/O field, 2092
  Graphic view, 2090, 2813
  I/O field, 2088
  IO field, 2811
  Line, 2097
  Recipe view, 2105
  Rectangle, 2104
  Simple recipe view, 2242, 2840
  Simple user view, 2823
  Switch, 2106, 2818
  Symbolic I/O field, 2109
  Text field, 2110
  Trend view, 2095, 2815
  User view, 2085
Layout of the force table, 1159
LED ACK, 2833
LED status
  Reading out with LED, 1745
Length
  Area pointer, 2408, 2410
Less than or equal to, 1214
Library, 2070
  Adding and using a block, 848
  Basics, 272
  Buttons and Switches, 2073
  Copy template, 2072
  Copying a library object, 2072, 2077
  Deleting a library object, 2077
  Global libraries, 285
  opening, 2082
  Project library, 276
  Save, 2081

Storing an object, 2078
Task card, 273
Type, 2072
Types, 272
Using filter view, 285, 302
Using the element and part views, 276
Library object, 2070, 2071
Inserting, 2079
License
Defective, 70
Defective license, 70
Starting without valid license, 66
License key, 68
Working with license keys, 68
License Manager Panel plug-in, 67
Licenses, 71
Limit, 2367, 2368
LIMIT, 1305, 1475, 1592
Limit frequencies of pulse outputs, 2930
Limit range
Tag, 2132
Limit value, 1305, 1475, 1592, 2168
Tag, 2133
Limit value check, 2804, 2806
Limit value exceeded
Analog alarm, 2133
Line, 2097
design, 2001
Line end, 2001, 2097
Line start, 2097
Line break, 609
Line end
Line, 2097
Line start
Line, 2097
Linear programming, 694
LinearScaling, 2320
Link folder
External graphic, 2006
Linked objects
Copying, 2763, 2764
Linking HMI tags, 157
LLDP (Link Layer Discovery Protocol), 514, 2431, 2483
LN, 1309, 1480, 1596
Load
from a device, 39
In programming device/PC, 524
SIMATIC PC station, 57
Load memory, 535, 1117
Displaying, 1120
Loading
Downloading blocks to a memory card, 1083
Downloading blocks to an accessible device, 1081
Downloading project data to the device, 227
Downloading to a memory card, 229
from a device, 230
General information, 226
Uploading blocks from a memory card, 1085
Loading projects
with connected HMI device, 2768
Local assignment
Of a function key, 2051, 2055
Local data bit (L), 535
Local data byte (LB), 535
Local data double word (LD), 535
Local data word (LW), 535
Local error handling, 1011
Error information, 1011
Instructions, 1011
Priorities, 1012
Local tag, 875
Local time, 548
Calculating with SET_TIMEZONE, 1675
Reading out with RD_LOC_T, 1674
Log
Language switching, 2747
Runtime language, 2747
Log file
Modem alarms, 3139
Log language, 2747
Logarithm, 1309, 1480, 1596
Logic operation
AND, 1369, 1541
EXCLUSIVE OR, 1372, 1545
OR, 1370, 1543
Logic path
Deleting, 974
Inserting, 973
Use, 972
Logoff, 2297
Users, 2825
LogOff, 2297
Logoff time, 2822
Changing, 2263
Changing in runtime, 2270
Logon, 2302
Logon failed, 2272
Reporting, 2276
Users, 2271
Logon dialog
Configuring access protection, 2275
Logon for web server, 437
Logon web server, 437
Loop, 1631, 1636, 1638, 1639, 1641, 1642

Loop-in alarm
  Configured, 2178
Loop-In-Alarm, 2367
Low free storage space, 2371
LREAL, 756, 794, 812

M

managing
  Graphic, 2005
Managing
  User group, 2263
  Users, 2262
  Users in runtime, 2269
Manual fragment, 458
Manuals, 322
Math functions
  CALCULATE, 920, 958
MAX, 1303, 1473, 1590
Maximum, 1303, 1473, 1590
Maximum cycle time, 531, 552, 1362, 1534
Maximum length
  Tag, 58
Maximum load memory available
  Displaying, 1120
Maximum number of controllable drives, 2930
MB_SLAVE, 40
MC_ChangeDynamic:Instruction, 1846
MC_ChangeDynamic:Parameter, 1846
MC_CommandTable:Instruction, 1844
MC_CommandTable:Parameter, 1845
MC_Halt:Function chart, 1828
MC_Halt:Instruction, 1826
MC_Halt:Parameter, 1827
MC_Home:Instruction, 1822
MC_Home:Parameter, 1824
MC_MoveAbsolute:Function chart, 1831
MC_MoveAbsolute:Instruction, 1829
MC_MoveAbsolute:Parameter, 1830
MC_MoveJog:Function chart, 1843
MC_MoveJog:Instruction, 1840
MC_MoveJog:Parameter, 1841
MC_MoveRelative:Function chart, 1835
MC_MoveRelative:Instruction, 1832
MC_MoveRelative:Parameter, 1833
MC_MoveVelocity:Function chart, 1839
MC_MoveVelocity:Instruction, 1836
MC_MoveVelocity:Parameter, 1837
MC_Power:Function chart, 1820
MC_Power:Instruction, 1816
MC_Power:Parameter, 1817
MC_Reset:Instruction, 1821
MC_Reset:Parameter, 1821
Meaning of the columns in the force table, 1159
Measuring window (GATE), 605
Memory area
  Load memory, 535
  Retentive memory areas, 537
  Work memory, 535
Memory bit (M), 535
Memory byte (MB), 535
Memory card, 533
  Accessing, 270
  Adding a card reader, 269
  Introduction, 269
  Removal/insertion, 36
  Showing properties, 271
Memory double word (MD), 535
Memory requirements
  Recipe, 2854
Memory reset, 532
  Making, 671
Memory word (MW), 535
Menu command
  Simple recipe view, 2243, 2841
Message
  Sending, 568
  Specifying the end, 569
  Specifying the start, 568
Micro, 2641
MicroLogix, 2569, 2595
Migrated project
  Compiling, 119
Migrating projects
  Procedure, 92
  Requirements, 92
Migration, 105, 107, 108, 110
  Adaptations beforehand, 121
  Alarm groups, 126
  Allen-Bradley DF1 data types, 140
  Allen-Bradley Ethernet IP data types, 141
  Basics, 115
  Data types Modicon Modbus, 143
  Data types Modicon Modbus TCP/IP, 143
  Data types Omron Hostlink/Multilink, 144
  Data types SIMATIC S7 200, 145
  Data types SIMATIC S7 300/400, 146
  Displaying a log file, 95
  Displaying history, 94
  Excluding the hardware configuration, 88
  Font settings, 124
  Global library, 132
  Integrated project, 150
  Introduction, 114

Introduction to migration, 87
Migrating an integrated project, 135, 150
Migrating projects, 117, 136
Mitsubishi FX data types, 142
of external tags, 139
of languages, 129
of project text, 129
Project library, 131
Recipe data, 132
Runtime data, 132
Supported HMI devices, 120
Supported products, 87
Tags, 125
Text, 129
The migration process, 87
User administration, 132
Migration of STEP 7 projects
CBA for IO controllers, 101
I slaves and I devices, 102
Introduction, 96
Isochronous mode with PROFINET IO, 101
Media redundancy, 101
Messages, 104
Migratable types of connection, 103
Multiproject, 104
Names of modules and racks, 100
PROFIBUS isochronous mode, 102
Requirements, 98
Requirements for the project, 98
Scope of migration, 96
Shared devices, 101
Station names, 100
System error messages, 105
Transfer areas, 102
Update groups PROFINET IO, 102
Migration tool, 116
Connecting the hardware configuration, 90
Creating a migration file, 91
Distribution and sources, 84
Removing, 85
System requirements, 84
Using the migration tool, 89
MIN, 1301, 1471, 1588
Minimum, 1301, 1471, 1588
Minimum cycle time, 531, 552
Mitsubishi, 2561
Address, 2615
Analog alarm, 2628
Basic Panel, 2413
Communication drivers, 2603
Connection interruption, 52
FX, 2603
TCP/IP, 2603
Mitsubishi FX
Configuring a connection, 2617
Connection, 2617, 2621
Connection parameters, 2618
CPU type, 2624
Data type, 2623
Migrating data types, 142
Mitsubishi MC TCP/IP
Configuring a connection, 2604
Connection, 2604, 2608
Connection parameters, 2606
CPU type, 2614
Data type, 2567, 2613
MOD, 1295, 1464
Modbus communication
as a slave with MB_SLAVE, 1936
as Modbus master with MB_MASTER, 1929
Configuring a port with MB_COMM_LOAD, 1926
Mode, 2107
Graphic I/O field, 2092
I/O field, 2088
Modem
Local, 3097
Problem, 3135
Remote, 3098
Modem alarms
Log file, 3139
Modem connection cannot be established, 3121
Modem connection is closed, 3122
Modem connection is interrupted, 3138
Causes and solutions, 3138
Modem support, 3097
Modem types / media, 3097
Modems without plug-and-play, 3098
Modicon
Approved communication with Modbus RTU, 2647
Connection, 2647
Connection cable, 2647
Restrictions with Modbus RTU, 2648
Modicon M340, 2641
Modicon Modbus, 2561
Basic Panel, 2413
Communication drivers, 2631
Migrating data types, 143
RTU, 2631
TCP, 2631
Modicon Modbus RTU, 2652
Configuring a connection, 2643
Connection, 2643
Connection parameters, 2645
Data type, 2651

Modicon Modbus TCP, 2641
Configuring a connection, 2631
Connection, 2631, 2636
Connection parameters, 2633
Data type, 2639
Modicon Modbus TCP/IP
Migrating data types, 143
Modicon MODBUS TCP/IP
Change word order, 2635
Modify
High-speed counters, 1848
Modify value
Permitted operands, 1141
Modifying mode, 1145
Module
Addressing, 430
Configuring and assigning, 36
Copying, 342
Deleting, 341
Determine diagnostics status, 664
Inserting, 36, 339, 340
Moving, 344
Outputting status information with ModuleStates, 1748
Outputting status of an IO system with DeviceStates, 1747
Removing, 36
Replacing, 345
Select, 330
Time of day on a module, 672
Module addressing, 540
Module arrangement, 327
Module replacement, 153
Module titles, 329
Module version
Updating, 347
Momentum, 2641, 2652
Monitor, 511, 2428, 2481
Monitoring
"Monitor all" command,
"Monitor now" command,
"Monitor once and now" command,
Monitoring and modifying modes, 1145
Monitoring mode, 1145
Monitoring window, 612
Mono-master system, 467
Motion Control CPU S7-1200
Guidelines, 2938
Move
Screen, 1975
MOVE, 1323, 1495
Move operation
Block, 1501
Block uninterruptible, 1503
Value, 1495
Move view
Keyboard shortcuts, 324
Overview navigation, 308, 311, 314
MOVE_BLK, 1330, 1501, 1607
Moving
Hardware components, 344
MPI, 2387
Addressing, 2501
Connection, 2524
HMI connection, 2493
Network, 2389
Network architecture, 2390
Parameters, 2495, 2497, 2499
S7 200, 2530, 2539
S7 300/400, 2492, 2493
MPI address
S7 300, 2501
S7 400, 2502
MPI communication
S7 300/400, 2492
MPI parameters
S7 300/400, 2495, 2497, 2499
MUL, 1292, 1461
Multi-instance, 728
Correcting the call type, 919, 957
Declaring, 888
Definition, 706
Multiple selection, 330, 1992
Multiplex, 1380, 1552, 1655
Multiplexing, 2139
Address multiplexing, 2144
Address multiplexing Allen-Bradley Ethernet IP, 2573
with absolute addresses, 2144
with symbolic addresses, 2144
Multiply, 1292, 1461
Multi-point connection
Allen-Bradley DF1, 2588, 2589
MUX, 1380, 1552, 1655

N

N, 1239, 1241, 1409, 1412
N_TRIG, 1244, 1414
Name
Array element, 50
Names of alarm classes
Change through migration, 127
NEG, 1296, 1466

Negate, 1296, 1466
Nesting depth, 703
Network, 2381
    Closing, 905, 945
    Copying, 904, 944
    Deleting, 905, 945
    Entering a comment, 907, 947
    Entering a title, 906, 946
    Ethernet, 2388
    Inserting, 902, 904, 942, 944
    MPI, 2389
    Navigating blocks, 907, 947
    Opening, 905, 945
    PPI, 2390
    PROFIBUS, 2389
    PROFINET, 2388
    Selecting, 903, 943
    Using, 902, 942
Network access
    Opening the properties, 3073
Network architecture
    PPI, 2391
Network comments
    Hiding, 870
    Showing, 870
Network drive, 32
Network editor
    Components, 305
    Function, 305
    Hardware catalog, 320
    Inspector window, 318
Network overview
    Basic functions, 352
    Basic functions for editing the network overview table, 353
Network view, 2391
    Adding device, 337
    Hardware and network editor, 307
Networking
    Communication partners, 2392
    Devices, 2391
Networking devices
    Basics of configuring networks, 348
    Editing interface parameters, 355
    Editing network parameters, 354
    Networking several interfaces at the same time, 350
    Networking with an existing subnet, 352
    Networking without an existing subnet, 350
    Networks within a project, 348
    Requirements, 354
    Types of communication, 348
Networking in the device view
    Networking options, 353
    Procedure, 353, 354
Non-integrated
    Connection, 2397
Non-licensed mode
    Engineering system, 66
    HMI devices, 68
NORM_X, 1350, 1522, 1630
Normalize, 1350, 1522, 1630
Normally closed contact, 1204, 1206, 1209, 1227
Normally open contact, 1203, 1204, 1206, 1209, 1226
NOT, 1228
NOT_OK, 1285, 1454

O

O, 3065
OB 1, 574
OB 80, 579
OB 82, 580
object
    Group, 2010
    Paste, 2001
    Rotate, 1999
Object
    Arrange, 1986, 1997
    Assigning to a layer, 2065
    Availability for Basic Panels, 2083
    Creating a new OLE object, 1987
    Creating an OLE object from a file, 1987
    Customizing default properties, 1987
    Deleting, 1989
    Design, 2001
    Dynamization, 2038
    Dynamization of the appearance, 2041
    Dynamize direct movement, 2044
    editing within a group, 2014
    Evenly distributing, 1995
    Flip, 1986
    Flush alignment, 1995
    Grouping, 2010
    Inserting, 1987
    Inserting of the same type, 1987
    Inserting several times, 2001
    Multiplying, 2001
    outside the area, 1998
    Paste, 1986
    Repositioning and resizing multiple objects, 1987, 2003
    Repositioning objects, 1986, 1990
    Resize, 1992
    Resizing, 1986

Rotate, 1986, 1999
Select multiple objects, 1992
Selecting multiple objects, 1987
Storing in a library, 2078
Tab sequence, 1987
Object group
Animation, 2047
Editing an object within a group, 2014
Removing an object, 2013
Object list
Tag, 2123
Object names
Uniqueness, 115
Object size
Harmonizing, 1996
Obtaining user rights
Logging on to the operating system with administrator privileges, 199
With Windows user account control, 198
OF, 1634
Off delay, 1249, 1256, 1420, 1571
OK, 1284, 1453
OLE object
create from a file, 1987
Recreate, 1987
Save in Graphics, 2007
Storing in the image browser, 2739
Omron, 2561
Analog alarm, 2673
Basic Panel, 2413
Communication drivers, 2657
Host Link, 2657
Omron Host Link, 2669
Address, 2670
Configuring a connection, 2658
Connection, 2658, 2662
Connection cable, 2662
Connection parameters, 2660
Data type, 2668
Omron Hostlink/Multilink
Migrating data types, 144
On delay, 1247, 1255, 1418, 1568
On demand, 2136
Ones complement, 1374, 1546
Online access, 3069
Online and Diagnostics view, 647
Online connection, 3065
multiple TIA portal instances, 36
Online displays
Orange color, 654
Online mode, 3065
Canceling an online connection, 3075
Display of the online mode, 3067
Displaying accessible devices on an interface, 3071
Displaying all accessible devices as a list, 3071
Go online, 3074
Standby or hibernation of the programming device / PC, 3065
Online operation, 36
Online Tools, 648, 656
OP 73
Baud rate on PROFIBUS, 2468, 2528, 2530, 2537, 2539, 2541
OP 77A
Baud rate on PROFIBUS, 2468, 2528, 2530, 2537, 2539, 2541
OP73
Loading a project, 56
OP77A
Loading a project, 56
OPC
Basic Panel, 2413
Open
Global library, 2082
Open User Communication
Ability to read back, 408
Changing IP configuration parameters with T_CONFIG, 1887
Changing parameter values, 407
Connection description, 403
Connection establishment, 392
Connection parameter assignment, 392, 393
Connection parameters, 395
Creating a connection, 398
Deleting an interconnection, 400
Establishing a communications connection with TCON, 1868
Establishing a connection and reading data with TRCV_C, 1863
Establishing a connection and sending data with TSEND_C, 1859
General, 391
Instructions, 391
Port numbers, 407
Protocols used, 401
Receiving data via UDP with TURCV, 1884
Receiving data with TRCV, 1876
Sending data via UDP with TUSEND, 1881
Sending data with TSEND, 1873
Starting connection parameter assignment, 397
TCON_Param, 404
Terminating a communications connection with TDISCON, 1871
TSAP, 408

Opening a force table, 1161
Opening a watch table, 1138
Opening phone books, 3124
Operand, 716, 717, 718, 719, 724, 725, 726, 728, 729, 730, 731, 732, 733, 734, 735, 738, 739
  Inserting, 930, 968
Operand area, 535
Operating behavior
  Enabling system memory, 571
  Using clock memory, 571
  Using time-of-day functions, 554
Operating element
  Availability for Basic Panels, 2083
Operating mode
  Introduction, 525
  RUN, 531
  STARTUP, 527
  STOP, 532
  Transitions, 526
Operating recipes
  Creating a recipe data record, 2245
  Creating recipe data records, 2246
  Read recipe data record, 2247
  Transfer data record, 2247
Operating system
  Asian language setting, 2725
  Setting to Western, 2725
  Updating, 2781
Operating system, 693
Operation
  Alarm indicator, 2194
  Button, 2817
  Date/time field, 2809
  Graphic IO field, 2814
  Graphic view, 2813
  IO field, 2811
  simple alarm view, 2832
  Simple alarm view, 2193
  Simple recipe view, 2242, 2840
  Simple user view, 2823
  Switch, 2819
  Symbolic IO field, 2820
  Trend view, 2816
Operator control
  Trend view, 2816
Operator input in the recipe view
  Transferring data, 2452, 2511, 2550, 2687
Operators, 980
Optimized access, 700
Optimizing the configuration
  Allen-Bradley DF1, 2578, 2598, 2616, 2625, 2642, 2652, 2671
Option handling, 482
Options for establishing the connection
  S7-1200 CPUs, 3127
  S7-300/400 CPUs, 3127
OR, 1370, 1394, 1395, 1543
  Logic operation, 1211
Organization block (OB)
  Assigning event with ATTACH, 1729
  Canceling event assignment with DETACH, 1730
  Creating, 844
  Cyclic interrupt OB, querying parameters with QRY_CINT, 1733
  Cyclic interrupt OB, setting parameters with SET_CINT, 1731
  exporting to an external source file, 1049
  Function, 696
  Start information, 696
Other PLCs
  Addressing, 2562
  Data types, 2562
  Special features, 2562
OUT_RANGE, 1282, 1451
Outgoing, 2161, 2366
Output
  Inserting, 927, 965
  remove, 928, 966
  Resetting, 1204, 1211, 1212
  Setting, 1204, 1211, 1212
Output (O), 535
Output byte (QB), 535
Output coil, 1209
Output double word (QD), 535
Output fields
  Multiplex tag, 51
Output word (QW), 535
Overflow, 609, 2370
Overlay icon, 652
Overlaying
  Tag, 40, 731
Overlaying tags, 887, 1022, 1043
Overview
  using the display formats in the force table, 1165
  via the display formats in the watch table, 1143
Overview navigation, 308, 311, 314
Overview of the cross-reference list, 1124
Overview of TS Adapters that can be used, 3100
Overview window, 44, 190
  Comparing objects, 192
  Showing and hiding columns, 193

P

P, 1204, 1212, 1237, 1240, 1408, 1410
P_TRIG, 1243, 1413
PageDown, 2337
PageUp, 2336
Pane mode, 188
Panning, 324
Parallel installation, 64
Parameter
  For CPU, 548
  Hidden parameters, 931, 969
Parameter transfer, 707, 709, 710, 711, 712, 713, 714, 715
  Rules, 708
Parameter types, 774
Parameters
  Connection, 2564, 2582, 2606, 2618, 2633, 2645, 2660
  MPI, 2495, 2497, 2499
  PROFIBUS, 2434, 2436, 2438, 2487, 2488, 2490
  S7 200, 2465, 2466, 2468, 2525, 2526, 2528, 2530, 2535, 2536, 2537, 2539, 2540
Password, 2822
  Backing up, 2822
  Changing, 2263
  Changing in runtime, 2270
  for function key, 2058
  Hierarchy levels, 2256
  Password aging, 2257
  Password complexity, 2257
  Restoring, 2822
Password list, 2822
Password protection, 437, 548, 572
  Protection concept, 234
  Revoking access rights, 235
Pasting
  Adjusting screen size, 2761
  Color, 2761
  Font, 2761
  Function key, 2761
  Invalid object, 2760
  Principle, 2759
Performance features
  Engineering system, 2848
  HMI device, 2850
Performance in telephone networks, 3097
Permitted characters
  User view, 58
Permitted operands for the force table, 1163
Permitted operands for the watch table, 1141
PG/PC interface
  Connecting online, 3074
  Modifiable MPI interface parameters, 3085
  Modifiable PROFIBUS interface parameters, 3085
Phone book
  Printing, 3126
PID_3Step
  Input parameters, 1794
  Instruction, 1788
  Static variables, 1797
PID_Compact
  Input parameters, 1775
  Instruction, 1771
  Static variables, 1777
Pin assignment
  6XV1440 - 2P for Mitsubishi PG protocol, 2622
  Allen-Bradley cable 1747-CP3, 2592
  Allen-Bradley cable 1761-CBL-PM02, 2593
  Allen-Bradley cable 1784-CP10, 2591
  Multipoint cable 1, 2664
  Multipoint cable 2, 2665
  Point-to-point cable 2, 2650
  Point-to-point cable PP1 for Omron, 2666
  Point-to-point cable PP2 for Omron, 2667
PLC
  Connecting, 2566, 2608, 2621, 2636, 2662
  MPI parameters, 2499
  PROFIBUS parameters, 2438, 2490
  PROFINET parameters, 2424, 2477
  Reading recipe data record, 2846
  Tag, 2139
  Transferring recipe data record, 2847
PLC alarm
  Creating, 1187
  Deleting, 1189
  Editing, 1188
PLC data type
  Addressing, 729
  Creating, 1038
  Declaration in data blocks, 1022
  Declaration in the block interface, 886
  Declaration table for PLC data types, 1037
  Declaring, 1042, 1043
  Declaring ARRAY, 1040
  Declaring STRUCT, 1041
  Definition, 775, 1036
  Deleting, 1039
  Offline/offline comparison, 1055
  Overlaying tags, 1043
  Programming the structure, 1039
  Tag properties, 1044
PLC tag
  Automatically filling in cells, 842
  Carrying out offline/offline comparisons, 1057

Comparing a PLC tag table, 1055
Comparison, 1052
Comparison editor, 1059
Copying, 841
Declaring, 831, 832
Definition, 717
Filtering the comparison editor view, 1062
Importing and exporting, 1200, 1201, 1202
Monitor, 840
Offline/offline comparison, 1055
Permissible addresses and data types, 831
PLC tag table, 824, 825, 827, 828
Properties, 837
reconnecting, 2131
Retentive behavior, 834
Rules, 831
Sorting rows, 842
Updating comparison results, 1063
PLC tags, 444, 445, 447
PLC5, 2595
PN/PN couplers
Grouping, 520
Linking Ethernet subnets, 520
Pointer
ANY, 771
POINTER, 769
VARIANT, 773
POINTER, 769
Pointer name
Area pointer, 2408
Point-to-point communication (PtP), 561
Freeport protocol, 561
Point-to-point connection
Allen-Bradley DF1, 2587
Configuring communication parameters with PORT_CFG, 1895
Configuring receive parameters with RCV_CFG, 1900
Configuring serial transmission parameters with SEND_CFG, 1898
Deleting receive buffer with RCV_RST, 1910
Enable receiving with RCV_PTP, 1908
Initiating data transfer with SEND_PTP, 1906
Querying the signal state with SGN_GET, 1911
Setting output signals with SGN_SET, 1912
Port
Renaming, 425
Port numbers, 407
Port options, 512, 2429, 2482
Enable autonegation, 512, 2429, 2482
Monitor, 511, 2428, 2481
Transmission medium/duplex, 511
Transmission rate / duplex, 2428, 2481
Portal view, 167
Position
an object, 1986
Editing multiple objects, 2003
Of objects, 1990
Possible cause of error
Transferring data, 2455, 2515, 2554, 2690
PPI, 2387
Network, 2390
Network architecture, 2391
S7 200, 2540
Predefined alarm classes, 2163
Premium, 2641
Preparing migration, 100
Press, 2365
Press key, 2370
PressButton, 2333
Print scope, 237
Printable contents, 237
Printing
Changing the settings, 238
Creating a cover page, 243
Creating borders, 242
Documentation function, 235
Elements in the library, 241
Elements in the project tree, 241
Non-printable contents, 237
Non-printable objects, 237
Print contents, 240
Printing device view, 315
Printing network view, 315
Process a cover page, 244
Process borders, 244
Specify layout, 239
Specify print layout, 239
Specifying the print area, 249
Structure of printout, 236
Use borders, 241
Use borders and cover pages, 236
Use cover page, 239
Printing a project, 255
Printing of alarms
Configuring, 2182
Printing phone books, 3126
Priorities
Local error handling, 1012
Process control phase, 2787
Process data area
Module addressing, 540
Module start address, 539
Process image, 540

Basics, 537
I/O access error, 538
Updating, 538
Product support, 322
Enabling, 323
PROFIBUS, 362, 2387, 2440, 2491
Connection with PROFINET, 490
ET 200S in DPV1 mode, 484
HMI connection, 2432, 2485
Network, 2389
OP 73, 2468, 2528, 2530, 2537, 2539, 2541
OP 77A, 2468, 2528, 2530, 2537, 2539, 2541
Parameters, 2434, 2436, 2438, 2487, 2488, 2490
S7 1200, 2432
S7 200, 2468, 2528, 2537
S7 300/400, 2484, 2485
Standard, 2440, 2491
Universal, 2440, 2491
PROFIBUS cable configuration
Optical ring, 364
PROFIBUS communication
S7 1200, 2432
S7 300/400, 2484
PROFIBUS DP, 468, 2389
Connection, 2521
PROFIBUS parameters
S7 1200, 2434, 2436, 2438
S7 300/400, 2487, 2488, 2490
PROFIBUS profile, 362, 2440, 2491
Different profiles on the same subnet, 362, 2440, 2491
Effect on the transmission rate, 362, 2440, 2491
Meaning of profiles, 363, 2440, 2492
Profile, 362
PROFINET, 2387
Connection, 2520
Connection with PROFIBUS, 490
Device replacement without exchangeable medium, 514
HMI connection, 2418, 2419, 2471, 2472
S7 1200, 2418
S7 300/400, 2471
PROFINET device name, 499
PROFINET interface, 548, 659
PROFINET IO, 488, 493, 495, 504, 506
Port options, 511, 2428, 2481
PROFINET IO device
Assigning a name, 675
Name assignment from the project tree, 676
Name assignment from within the project context, 677
Name assignment via the "Online" menu,
PROFINET IO devices
Reading consistent data with DPRD_DAT, 1721
Writing consistent data with DPWR_DAT, 1723
PROFINET IO system, 505
PROFINET parameters
HMI connection, 2421, 2474
HMI device, 2422, 2475
PLC, 2424, 2477
S7 1200, 2421, 2424
S7 300/400, 2474, 2475, 2477
PROFINET RT class, 489
Program control operations, 1362, 1534, 1645
Program editor
"Instructions" task card,
Block interface, 865
Enlarging the programming window, 869
Favorites, 865
Function, 863
General settings, 162, 164, 165, 874
Programming window, 865
Structure, 864
Testing task card, 866
Toolbar, 865
Using the keyboard, 867
Program execution
Cyclic, 574
Program information
Displaying, 1093
In the assignment list, 1094
In the call structure, 1103
In the dependency structure, 1111
in the tab, 1117
Views, 1093
Program resources, 844
Program status
Editing blocks, 1132
FBD, 1129
Function, 1128
LAD, 1128
Modify tag, 1133
SCL, 1130
Switching off, 1131
Switching on, 1131
Programming
Linear, 694
Structured, 695
Programming error
System reaction, 37
Programming language
Changeover, 1009
FBD, 939
LAD, 898

Rules for changing, 1008
Programming window
Customizing, 985
SCL, 984
Project
Basics, 209
Closing, 213
Comparing data, 233
compile (overview), 2766
Compiling, 2767
Compiling project data, 224
Creating, 210
Deleting, 214
Download, 2770, 2788
download (overview), 2766
Migrating, 117, 136
multilingual, 2729
multiple connections, 2447, 2506
Multiple connections, 2546
Opening, 210
Saving, 213
Showing properties, 212
Simulation with a tag simulator, 2773
Project data
Compiling, 224
Downloading to a memory card, 229
Export alarm, 2698
Export text list, 2711
Exporting a recipe, 2694
Exporting tag, 2706
Importing alarms, 2699
Information on compiling, 224
Information on loading, 226
Loading, 227
Loading to an accessible device, 228
Uploading from a device, 230
Project ID, 2386
Project language, 2723
Activate, 2727
Disabling, 2727
Project languages
Changing the editing language, 217
Specifying project languages, 217
System texts, 215, 2730
Use, 215
User texts, 215, 2730
Project library, 44, 2070
Adding a copy template, 277
Adding types, 278
Basics, 276
Creating folders, 277
Editing elements, 282
Migrating, 131
Removing elements, 284
Using filter view, 285, 302
Using master copies, 280
Using the element and part views, 276
Using types, 281
Project text
Exporting, 2716, 2734
Project texts
Application example, 222
Compiling project texts, 218
Displaying reference text, 219, 2733
Exporting all texts, 220
Exporting individual texts, 220
Importing, 221, 2718, 2736
Migrating, 129
Translating individual texts, 2731
Translation into project languages, 215, 2730
Project tree
Adding device, 336
Function, 171
Hiding, 870
Keyboard shortcuts, 195
Reducing automatically, 178
Showing, 870
Project view, 169
Properties (CPU), 548
ProSave, 2777
Installation, 55
Protect
Function key with password, 2058
Protection concept
Introduction, 234
Revoking access rights, 235
Protection level, 572
Revoking access rights, 235
PT, 1431
Pulse, 1245, 1253, 1416, 1565
Pulse and direction output, 2931
Pulse generator
Compiling hardware, 39
Enabling/disabling with CTRL_PWM, 1755
Pulse interface:Principle, 2933
Pulse stretching, 612

Q

Q address, 430
Q series, 2614
Quantum, 2641

R

R, 362, 1203, 1204, 1211, 1212, 1231, 1401
Racks, 329
  Inserting a module, 339
Radius, 2094
Range
  Value outside range, 1282, 1451
  Value within range, 1281, 1450
Range (... - ...)
  Graphics list, 2033
RE_TRIGR, 1362, 1534, 1646
Read continuously
  Tag, 2136
READ_DBL, 41
Reading
  from a data block in load memory with READ_DBL, 1767
Reading data
  From DP standard slaves/PROFINET IO devices with DPRD_DAT, 1721
  From remote CPU with GET, 1854
Reading out
  LED status with LED, 1745
  Local time with RD_LOC_T, 1674
  Recipe data record, 2846
Reading out a diagnostics buffer, 665
Reading tags, 445
Read-only, 572
REAL, 755, 793, 811
Recipe, 2219, 2221, 2222, 2697, 2837
  Application example, 2221
  Basics, 2219, 2221
  Control, 2838
  Creating new, 2232
  Data flow, 2839
  Data record, 2222, 2837
  Entry, 2222
  Exporting, 2694
  Importing, 2696
  Memory requirements, 2854
  Use, 2221
  Use of text lists, 2227
Recipe data
  Migration, 132
Recipe data record
  Create on the HMI device, 2246
  Creating, 2843
  Creating new, 2232
  Creating on the HMI device, 2245
  Deleting, 2845
  Editing, 2844
  Reading from PLC, 2846
  Synchronize with PLC, 2844
  Transferring the project, 60
  Transferring to PLC, 2847
  Use of text lists, 2227
Recipe element
  Creating new, 2232
Recipe list, 2229
Recipe view, 2228
  Application, 2105
  Behavior with screen change, 2229
  Configuring, 2240
  Control element, 2105
  Layout, 2105
  Operation using the function key, 2230
  Recipe data record, 2223, 2230
  Simple, 2229
  Updating, 2229
RecipeViewBack, 2331
RecipeViewClearDataRecord, 2327
RecipeViewGetDataRecordFromPLC, 2326
RecipeViewMenu, 2327
RecipeViewNewDataRecord, 2326
RecipeViewOpen, 2328
RecipeViewRenameDataRecord, 2330
RecipeViewSaveAsDataRecord, 2329
RecipeViewSaveDataRecord, 2329
RecipeViewSetDataRecordToPLC, 2328
RecipeViewShowOperatorNotes, 2330
Recommissioning
  HMI device, 2788
Reconnecting
  Tag, 2131
Reconnecting a PLC tag, 2131
Recording a log file for the modem, 3135
Rectangle
  Configuring, 2020
  Inserting, 2020
  Inserting and configuring, 2019
  Radius corners X, 2104
  Radius corners Y, 2104
Redoing actions
  Basics of redoing actions, 257
  Redoing actions, 260
Reference junction, 479, 594, 609
Reference language, 2723
  Selecting, 2728
Reference object
  Defining, 1993
release, 2367
ReleaseButton, 2334
Remainder of division, 1295, 1464
Remote connection cannot be established, 3120

Remote connection from the TS Adapter is not established, 3137
Remote connection to the TS Adapter is not established, 3136
Remote system access to a programming device/personal computer, 3127
Removal/insertion
  System reaction, 37
Remove
  Object from the group, 2013
Removing
  Folder link, 2006
  Formatting in the alarm text, 2177
Rename
  Screen, 1975
  Tag, 2127
  Template, 1981
REPEAT, 1639
Replacing
  Module, 345
Replacing devices
  Limitations on connections, 2751
  Principle, 2751
Report System Errors, 620
  Created error OBs, 623
Requirements for establishing a remote connection, 3119
Requirements for sending an SMS, 3131
Requirements for sending e-mails, 3132
  Required communication instructions, 3134
Reset
  Bit field, 1234, 1404
  IEC timer, 1259
  Operand, 1231, 1236, 1401, 1406
  Output, 1211
Reset output, 1203
Reset to factory settings, 2780
RESET_BF, 1234, 1404
ResetBit, 2331
ResetBitInTag, 2332
Resetting the MPI/PROFIBUS settings, 3093
Resetting the TCP/IP settings, 3084
Resource
  SIMATIC S7 1200, 54
Resources, 1117
  Introduction, 1117
  Load memory, 1117
  Retentive memory, 1117
  Retentive memory data, 1117
  Work memory, 1117
Resources tab
  Structure, 1119
Restore process
  Authorization, 67
  License key, 67, 69
Restoring, 2822
  Data of the HMI device, 2778, 2779
  HMI device data, 2794
Restoring data
  HMI device, 2778, 2779, 2794
Restrictions due to user rights, 197, 198
  Recognizing, 198
RET, 1360, 1532
Retentive behavior
  Bit memory, timers, counters, 834
Retentive bit memories
  Enabling the display, 1102
Retentive memory, 1117
Retentive on delay, 1209, 1217
Retentivity, 537, 700
  Bit memory, timers, counters, 834
  Block interface, 891
  Data block, 1025
  Data block, 1025
  PLC tag, 834
RETURN, 1644
RLO
  Invert, 1228, 1399
ROL, 1390, 1563, 1666
Root, square, 1595
ROR, 1388, 1561, 1664
Rotate
  Left, 1390, 1563, 1666
  Object, 1986
  Right, 1388, 1561, 1664
Round, 1342, 1344, 1345, 1347, 1514, 1516, 1517, 1519, 1625
ROUND, 1342, 1514, 1625
RS, 1236, 1406
RS-232/PIP multi-master cable, 566
RT, 1259
RT class, 489
Rules for configuring MPI networks
  Rules for assigning the MPI address, 357
Rules for network configuration for PROFIBUS networks
  Assigning device addresses, 357, 358
RUN, 531
Rung
  Deleting, 939
  Inserting, 938
  Use, 937
Runtime
  Execution of the function list, 2291

Mouse wheel, 59
Simulating, 2772
Start screen, 1976
Runtime data
Migration, 132
Runtime language, 2724, 2741, 2795
Font, 2746, 2797
Log, 2747
Order for language switching, 2744
Selecting, 2742, 2795
Runtime meters
Handling with RTM, 1677
Runtime settings
User administration, 2256
Runtime start
Basic Panel, 2771, 2792
Runtime Stop, 2369

S

S, 1203, 1204, 1211, 1212, 1232, 1402
S5TIME, 758
S7 1200
Basic Panel, 2413
Communication, 2417
HMI connection, 2419
PROFIBUS, 2432
PROFIBUS parameters, 2434, 2436, 2438
PROFINET, 2418
S7 1200 V1
Data types, 2460
S7 1200 V2
Data types, 2461
S7 200
Communication, 2532
Connection, 2532
Connection parameters, 2465, 2535
Data types, 2560
Ethernet, 2466, 2526, 2536
MPI, 2530, 2539
Parameters, 2465, 2466, 2468, 2526, 2528, 2530, 2535, 2536, 2537, 2539, 2540
PPI, 2540
PROFIBUS, 2468, 2528, 2537
S7 300
Basic Panel, 2413
Communication, 2470
MPI address, 2501
S7 300/400
Connection parameters, 2525
Data types, 2519
HMI connection, 2472
MPI, 2492, 2493
MPI parameters, 2495, 2497, 2499
Parameters, 2525
PROFIBUS, 2484, 2485
PROFIBUS parameters, 2487, 2488, 2490
PROFINET, 2471
PROFINET parameters, 2474, 2475, 2477
S7 400
Basic Panel, 2413
Communication, 2470
MPI address, 2502
S7 connection, 369
TSAP, 381
S7 CPU
Load memory, 535
Operand area, 535
Work memory, 535
S7-PCT, 522
Safety guidelines, 207
Safety instruction
Changing the recipe data record in the background, 2229
Switching tooltip, 2807
Safety precautions when forcing tags, 1158, 1173
Save
Global library, 2081
Saving a force table, 1162
Saving a watch table, 1139
Scale, 1348, 1520, 1628
SCALE_X, 1348, 1520, 1628
Scaling
Applying linear scaling to a tag, 2137, 2138
Scheduler, 2373
Deactivated task, 2377
Event trigger, 2377, 2380
Function list, 2376
Triggers, 2377
Work area, 2375
SCL, 974
Migration, 110
Programming window, 984
Restriction, 39
SCL instruction
Inserting, 992
Restriction, 39
Rules, 991
Selecting, 1005
SCL program
Inserting a block call, 995, 997, 998, 999, 1000, 1001
Inserting an instruction, 992
Inserting comments, 1003

Program status display, 1130
Restriction, 39
Selecting an instruction, 1005
Screen
Availability for specific devices, 1969
Copy, 1975
Copying, 2761
Creating, 1974
Deleting, 1975
Font, 1970
Inserting, 1975
Move, 1975
Rename, 1975
Use template, 1983
Working steps in creating, 1974
Zooming, 1973
Screen change, 2363
Screen keyboard
Alphanumerical, 2803, 2805
Key assignment, 2803, 2805
KTP400 Basic, 2802
KTP600, KTP1000, TP1500 Basic, 2805
Language behavior, 58
Numerical, 2804, 2806
Screen number, 2386
Screen object
Dynamic movement, 2043
Screen saver, 30
Screen template, 2053
ScreenObjectCursorDown, 2304
ScreenObjectCursorUp, 2304
ScreenObjectPageDown, 2306
ScreenObjectPageUp, 2305
Script
Updating tag value, 2118, 2403
SD card
Open force job, 30
Search
Hardware catalog, 328
Security system, 2821
Segment diagnostics
Graphic display, 684
Icons, 684
Text display, 684
SEL, 1378, 1550, 1654
Select, 1378, 1550, 1654
Multiple objects, 1992
Selecting
Multiple objects, 1987
Selection changed, 2362
Send clock cycle, 508, 509
Service & Support, 322
Enabling, 323
Servo motor, 2929
Set
Bit field, 1233, 1403
Operand, 1232, 1235, 1402, 1405
Output, 1211
Set limit value, 1305, 1475, 1592
Set output, 1203
SET_BF, 1233, 1403
SetBit, 2338
SetBitInTag, 2339
SetBitWhileKeyPressed, 2340
SetColorBackgroundLighting, 2341
SetConnectionMode, 2344
SetDeviceMode, 2337
SetLanguage, 2342
SetTag, 2343
Setting
Deadband, 2175
Delay time, 2175
Language, 2808
Languages in the operating system, 2724
View options for the assignment list, 1098
View options for the dependency structure, 1114
Setting AS-i network configuration parameters, 368
Setting ENO automatically, 982
Setting language, 2808
Setting parameters for the Ethernet interface, 3082
Adding a dynamic IP address, 3083
Connecting to a subnet, 3081
Deleting dynamic IP addresses, 3084
Modifiable parameters, 3079
Options in the parameter settings, 3079
Setting the communication load, 365
Setting the font, size and color, 985
Setting the mnemonics, 896
Setting the tab spacing, 985
Settings
Changing, 166, 875, 901, 942, 983
FBD, 901, 941
General, 162, 164, 165, 874
LAD, 901, 941
SCL, 982
Settings for the PG/PC interface, 3077
Automatic bus parameter detection, 3086
Setting parameters for the MPI interface, 3087
Setting parameters for the PROFIBUS interface, 3089
Shift
Left, 1386, 1559, 1662
Right, 1384, 1557, 1660
SHL, 1386, 1559, 1662

Show
  Limit lines on the bar, 2084
Show line numbers, 986
ShowAlarmWindow, 2353
Showing and hiding absolute operands, 986
Showing and hiding columns, 843, 895, 1033, 1047
ShowLogonDialog, 2352
ShowOperatorNotes, 2352
SHR, 1384, 1557, 1660
Shutting down Windows XP, 30
Sign, 1300, 1469
Signal board, 434
  Inserting, 434
SIMATIC PC station
  Load, 57
SIMATIC S7
  Communication partners, 2469, 2529, 2538, 2541
SIMATIC S7 1200
  Connection resource, 54
  Resource, 54
SIMATIC S7 200
  Migrating data types, 145
SIMATIC S7 300/400
  Migrating data types, 146
simple alarm view, 2832
  Control element, 2833
  Operation, 2832
  Use, 2832
Simple alarm view, 2192
  Control element, 2193
  Operation, 2193
  Use, 2192
simple alarm window
  Application, 2192
  Use, 2832
Simple recipe view, 2229
  Behavior, 2243, 2841
  Layout, 2242, 2840
  Menu command, 2243, 2841
  Operation, 2242, 2840
Simple user view, 2823
  Layout, 2823
  Operation, 2823
  Use, 2823
Simulate hardware, 304
Simulate modules, 304
Simulate software, 304
SimulateTag, 2346
Simulating
  Project with tag simulator, 2773
Simulating devices, 304
Simulation, 59
  PLC connection, 59
  Runtime, 2772
SIN, 1311, 1483, 1599
Sine, 1311, 1483, 1599
Single instance
  Correcting the call type, 919, 957
  Definition, 704
  Example, 705
SINT, 749, 787, 803
Size
  An object, 1986
  Editing multiple objects, 2003
SLC, 2595
Slot
  Racks, 329
  Select, 330
Slot rules, 327
  ET 200S, 478
Smoothing, 608
Softkey, 2051, 2055
  global, 2053
Software requirements at the plant end
  Communication instruction "AS_MAIL",
  Communication instruction "SMS_SEND",
  Communications instruction "PG_DIAL",
Software requirements for AS-AS remote link
  Communication instruction "AS_DIAL",
Software requirements for the programming device/personal computer
  Communications instruction "PG_DIAL",
Software-controlled data flow control, 564
Special characters
  Alarm text, 51
Special tag (web server), 448
SQR, 1594
SQR, 1594
SQRT, 1308, 1478, 1595
Square, 1306, 1477, 1594
Square root, 1308, 1478, 1595
SR, 1235, 1405
Stamping, 2001
Standard
  PROFIBUS, 2440, 2491
Standard device, 515
Standard slave, 488
Standby Mode, 30
Standby module
  Substitute value, 483
Start address, 430, 539
Start information, 696
Start screen, 1976
Start the online and diagnostics view, 655

Start the simulation, 304
Start value
  Tag, 2134, 2135
Starting removal, 82
Starting the migration tool, 90
Starting the topology view, 416
Startup
  Organization blocks, 573
STARTUP
  Function, 527
  STARTUP activities, 529
  Warm restart, 528
Startup OB
  Description, 573
Start-up parameters, 530
Startup program, 573
Startup type, 548
Station
  Copying, 342
  Deleting, 341
  Moving, 344
  Renaming, 425
Station name, 499
Status
  Module defective, 664
Status of the online connection, 680
Stepper motor, 2929
STL
  Migration, 107
STL source
  Generating blocks, 1051
STOP, 532
Stop forcing, 1178, 1179
StopRuntime, 2347
Storing
  External graphic, 2007
STP, 1362, 1534, 1645
STRING, 763, 798, 817
STRUCT
  Addressing, 729
  Declaration in global data blocks, 1021
  Declaration in the block interface, 885
  Declaring in a PLC data type, 1041
  Structure, 768
Structure
  Call structure, 1106
  of the cross-reference list, 1122, 2720
  Of the dependency structure, 1112
  Resources tab, 1119
  STRUCT, 768
Structure of the phone book, 3123
Structured programming, 695
SUB, 1291, 1460
Subnet mask, 495
Substitute value
  Standby module, 483
Subtract, 1291, 1460
Subtracting
  Subtracting times with T_SUB, 1669
Supported telephone networks and modems, 3096
  Restrictions, 3097
SWAP, 1339, 1511, 1615
Swap bytes, 1339
Switch, 2106, 2818
  Application, 2818
  Behavior, 2819
  Layout, 2818
  Operation, 2819
  Type, 2106
SWITCH, 1357, 1529
Switch off, 2371
Switch on, 2371
Switching between basic mode and expanded mode in the force table, 1160
Switching between basic mode and expanded mode in the watch table, 1135
Switching operating mode, 670
Switchover time, 567
Switchport, 512, 2429, 2482
Symbol
  Alarm classes, 2183
Symbolic addressing
  of a tag, 2119, 2404
Symbolic constants, 718
Symbolic I/O field, 2108
  Mode, 2109
  Text list, 2028, 2109
Symbolic IO field, 2820
  Application, 2820
  Behavior, 2820
  Operation, 2820
Symbolic programming
  Displaying absolute addresses, 896
Symbols
  In the assignment list, 1096
  In the call structure, 1105
  In the dependency structure, 1113
Sync domain, 514, 2431, 2483
Synchronization (custom web pages), 440
Synchronizing user-defined web pages, 1893
Synchronous
  Transferring data, 2510, 2555, 2685
Syntax errors
  Basics, 1007

Finding errors, 1008
Syntax for AWP commands, 444
System alarm, 2160
Meaning, 2197, 2201, 2207
System block
System blocks folder, 844
System diagnostics
Created blocks, 622
General settings, 633
Introduction, 620
Send alarms, 633
Settings for "PLC in STOP",
Settings for configuring alarms, 634
System event, 2160
Meaning, 2197, 2198, 2201, 2202, 2204, 2205, 2206, 2207, 2208, 2209, 2211, 2213, 2214
Parameters, 2196
System events
Configuring, 2168
System function
Application, 2286
Function list, 2288
In the function list, 2286
Usage, 2285
Use, 2286
System functions, 2285
AcknowledgeAlarm, 2325
ActivateCleanScreen, 2301
ActivatePreviousScreen, 2301
ActivateScreen, 2299
ActivateScreenByNumber, 2300
AdjustContrast, 2298
AlarmViewAcknowledgeAlarm, 2324
AlarmViewEditAlarm, 2323
AlarmViewShowOperatorNotes, 2325
AlarmViewUpdate, 2324
Available, 2294
CalibrateTouchScreen, 2312
ChangeConnection, 2350
ClearAlarmBuffer, 2321
ClearAlarmBufferProTool, 2322
DecreaseFocusedValue, 2348
DecreaseTag, 2349
EditAlarm, 2303
GetGroupNumber, 2319
GetPassword, 2319
GetUserName, 2318
GoToEnd, 2308
GoToHome, 2308
IncreaseFocusedValue, 2306
IncreaseTag, 2307
InvertBit, 2309
InvertBitInTag, 2310
InvertLinearScaling, 2311
LinearScaling, 2320
Logoff, 2297
Logon, 2302
PageDown, 2337
PageUp, 2336
PressButton, 2333
RecipeViewBack, 2331
RecipeViewClearDataRecord, 2327
RecipeViewGetDataRecordFromPLC, 2326
RecipeViewMenu, 2327
RecipeViewNewDataRecord, 2326
RecipeViewOpen, 2328
RecipeViewRenameDataRecord, 2330
RecipeViewSaveAsDataRecord, 2329
RecipeViewSaveDataRecord, 2329
RecipeViewSetDataRecordToPLC, 2328
RecipeViewShowOperatorNotes, 2330
ReleaseButton, 2334
ResetBit, 2331
ResetBitInTag, 2332
ScreenObjectCursorDown, 2304
ScreenObjectCursorUp, 2304
ScreenObjectPageDown, 2306
ScreenObjectPageUp, 2305
SetBit, 2338
SetBitInTag, 2339
SetBitWhileKeyPressed, 2340
SetConnectionMode, 2344
SetDeviceMode, 2337
SetLanguage, 2342
SetTag, 2343
ShowAlarmWindow, 2353
ShowLogonDialog, 2352
ShowOperatorNotes, 2352
SimulateTag, 2346
StopRuntime, 2347
TraceUserChange, 2348
TrendViewBackToBeginning, 2317
TrendViewCompress, 2314
TrendViewExtend, 2313
TrendViewRulerLeft, 2315
TrendViewRulerRight, 2315
TrendViewScrollBack, 2313
TrendViewScrollForward, 2312
TrendViewSetRulerMode, 2316
TrendViewStartStop, 2317
UpdateTag, 2302
System limits
HMI device, 2850
System memory, 548, 571

Diagnostics buffer, 539, 678
Operand areas, 535
process image input/output, 537
System requirement for WinCC Basic, 63
System requirements STEP 7 Basic, 62
System-defined text lists
Editing, 267
Modifying texts, 268
System-relevant information
Module information, 656
Vendor information, 656

T

Table
Connection, 2391
Tag
absolute addressing, 2118, 2403
Acquisition cycle, 2136, 2142, 2154
Acquisition mode, 2136
Addressing, 2122
Changing the PLC, 2139
Comment, 2124
Configuration, 2128
Configuring several tags, 2129
Connection to PLC, 2122
Copy, 2127
Creating external tags, 2122
Creating internal tags, 2124
Data type, 2124
Deleting, 2128
Displaying values, 2155
Event, 2141
Exporting, 2706
External tag, 2117, 2402
Importing, 2707
in Runtime, 2136
Index tag, 2139, 2140
Indirect addressing, 2139, 2140
Internal tag, 2121, 2405
Length, 2124
Limit range, 2132
Limit value, 2133
Linear scaling, 2137, 2138
Maximum length, 58
Multiplexing, 2139
Name, 2124
Object list, 2123
Output in alarm, 2176
Overlaying, 731, 1022
Overview, 717
PLC tag, 824
PLC tags and DB tags, 717
Read continuously, 2136
reconnecting, 2131
Rename, 2127
Start value, 2134, 2135
symbolic addressing, 2119, 2404
Update, 2136, 2142
Tag data
Structure for the import, 2708, 2714
Tag declaration
Automatically filling in cells, 842, 894, 1033, 1046
Based on a PLC data type, 1022
Block interface, 876
Declaring ARRAY, 884
Declaring PLC data type, 886
Declaring STRUCT, 885
Declaring tags, 881, 882, 887, 1022
Deleting a tag, 841, 894, 1032, 1046
Importing and exporting tags, 895, 1034
Inserting a table row, 840, 893, 1031, 1045
Inserting table rows at the end, 893, 1032, 1045
Multi-instance, 888
Overlaying tags, 887, 1022
Purpose of tag declaration, 875
Reserved key words, 719
Retentivity, 891
Showing and hiding columns, 843, 895, 1033, 1047
Sorting rows, 842
Tag properties, 890, 891, 1028, 1029, 1030
Updating the block interface, 888
Valid data types, 879
Tag import
Tag data structure, 2708, 2714
Tag list
Indirect addressing, 2139, 2140
Tag simulator, 2775
Tag table
Default, 2116, 2400
for HMI devices, 2116, 2400
user-defined, 2116, 2400
Tag value
Output, 2155
Tags
Basics, 2115, 2400
Migrating, 125
Migration, 139
Monitor all, 1147, 1169
Monitor now, 1170
Monitor once and now, 1148
Tan, 1602
TAN, 1314, 1486
Tangent, 1314, 1486, 1602

Task, 2374
Deactivate, 2377
Task card, 1970
Changing the pane mode, 188
Function, 186
Hardware catalog, 320
Reducing automatically, 178
Tools, 1983
Task Card
Online Tools, 656
TCI (Tool Calling Interface), 521
TCON_Param, 404
TCP
Characteristics, 402
Port numbers, 407
Technology object command table:Add new object, 2970
Technology object command table:Basic parameters, 2971
Technology object command table:Command table configuration, 2972
Technology object command table:Configuration window icons, 2971
Technology object command table:Configuring activate warnings, 2972
Technology object command table:Configuring duration, 2974
Technology object command table:Configuring position / travel path, 2973
Technology object command table:Configuring the command type, 2973
Technology object command table:Configuring the next step, 2975
Technology object command table:Configuring the step code, 2975
Technology object command table:Configuring use axis parameters of, 2972
Technology object command table:Configuring velocity, 2974
Technology object command table:Extended parameters, 2971
Technology object command table:General configuration, 2971
Technology object command table:Shortcut menu commands, 2975
Technology object command table:Tools, 2969
Technology object command table:Usage, 2969
Technology object, axis and command table:List of ErrorIDs and ErrorInfo, 3028
Technology objects
PID_3Step, 2907
PID_Compact, 2887
Telephone book at TeleService, 3096
Telephone networks which can be used, 3096
TeleService
AS_DIAL, 3129
Callback variants, 3116
Establish connection to AS, 3129
Global telephone book, 3096
Meaning of the icons, 3124
Transferring e-mail with TM_MAIL, 1958
TeleService callback options, 3118
TeleService functionality, 3095
TeleService telephone book, 3096
Temperature coefficient, 607
Temperature compensation, 594
template
Template, 1981
Template, 2051
Copy, 1981
Creating, 1980
Deleting, 1981
Global screen, 1980
Inserting, 1981
Move, 1981
Rename, 1981
Use in screen, 1983
Terminal modules and electronic modules, 592
Terminating the remote connection, 3120
Test options in the force table, 1156
Text
Button, 2108
Text field, 2110
Text field, 2110
Size, 2111
Text field length
Asian languages, 2799
Text list
Application, 2022
Bit (0, 1), 2026
Bit number (0 - 31), 2028
Creating, 2023
Exporting, 2711
Importing, 2713
Symbolic I/O field, 2028, 2109
Value/Range, 2025
Text lists
Introduction, 264
System-defined, 265
Use in recipe data records, 2227
User-defined, 265
Text lists for alarms, 1196
Texts
Migrating, 129

Third-party drivers
  Communication, 2561
  Special features, 2562
TIA Portal
  Exiting, 161
  Starting, 161
Time, 757, 758
  Converting UTC time to local time with SET_TIMEZONE, 1675
  Recording, 1258
TIME, 757, 794, 813
Time accumulator, 1251, 1422, 1574
Time duration, 1260
Time error interrupt OB, 579
Time expired, 2372
Time of day
  Setting the time of day in the Online and Diagnostics view, 672
Time stamp, 2167
Time stamping, 595
Time-delay interrupt
  Canceling with CAN_DINT, 1742
  Instructions, 1739
  Querying with QRY_DINT, 1743
  Starting with SRT_DINT, 1741
Time-delay interrupt OB, 575
Time-of-day, 548, 760, 761, 762
  Calculating local time with SET_TIMEZONE, 1675
  Reading time-of-day and date of CPU with RD_SYS_T, 1673
  Setting for CPU with WR_SYS_T, 1672
Time-of-day function
  Basics, 554
  Clock parameters, 555
  Reading the time of day, 555
  Setting the time of day, 555
  Time-of-day format, 554
Time-of-day interrupt
  Activating with ACT_TINT, 1737
  Canceling with CAN_TINT, 1736
  Querying with QRY_TINT, 1738
  Setting with SET_TINT, 1735
Time-of-day synchronization, 548
TIMER, 774
Timer instructions
  Retentive on delay, 1209, 1217
Times
  Adding with the instruction T_ADD, 1668
  Converting with T_CONV, 1668
  Determining difference with T_DIFF, 1670
  Subtracting with T_SUB, 1669
TM_MAIL, 31
TO_Axis_PTO, 2946
TOD, 760, 795, 814
TOF, 1249, 1256, 1420, 1571
Toggle
  Between runtime languages, 2798
  Between Runtime languages, 2741
TON, 1247, 1255, 1418, 1568
TONR, 1209, 1217, 1251, 1258, 1422, 1574
Tool Calling Interface (TCI), 521
Toolbar
  Order, 1986, 1997
Tools, 1983
Tooltip
  display, 2807
Topology discovery, 514, 2431, 2483
Topology view
  Adding device, 337
  Adopt devices identified online, 430
  Adopt port interconnections identified online, 429
  Compare offline/online, 420
  Configured topology, 416, 417
  Diagnostics status in the graphic view, 418
  Diagnostics status in the table view, 419
  Differences compared with the network view, 414
  Functions, 414
  Hardware and network editor, 313
  Interconnecting ports, 422
Touch operation
  IO field, 2812
TP, 1245, 1253, 1416, 1565
TP177A
  Loading a project, 56
TraceUserChange, 2348
Transfer card, 533
Transferring
  Recipe data record, 2847
Transferring data
  Area pointer, 2406, 2441, 2502, 2542, 2676
  Area pointer screen number, 2446, 2502, 2542, 2677
  Coordination area pointer, 2444, 2505, 2545, 2680
  Data record are pointer, 2684
  Data record area pointer, 2450, 2509, 2549
  Date/time area pointer, 2441, 2443, 2503, 2543, 2678
  Date/time PLC area pointer, 2504, 2544, 2679
  Job mailbox, 2453, 2512, 2551, 2688
  Job mailbox area pointer, 2448, 2507, 2547, 2682
  Operator input in the recipe view, 2452, 2511, 2550, 2687
  Possible cause of error, 2455, 2515, 2554, 2690
  Project ID area pointer, 2447, 2506, 2546, 2681

Triggering by means of a configured function, 2454, 2514, 2553, 2689
With synchronization, 2510, 2555, 2685
Without synchronization, 2451, 2510, 2555, 2685
Transferring the project
HMI device, 60
Recipe data record, 60
Translate
Editor, 2729
Transmission medium/duplex, 511
Transmission rate / duplex, 2428, 2481
Transparency
In graphic, 2004
transparent color, 2091
Display on panels, 2091
Display on the panel HMI, 2813
Trend view, 2095, 2815
Application, 2815
Button, 2095
Configuring for values from the PLC, 2155
Layout, 2815
Operation, 2816
Operator control, 2816
TrendViewBackToBeginning, 2317
TrendViewCompress, 2314
TrendViewExtend, 2313
TrendViewRulerLeft, 2315
TrendViewRulerRight, 2315
TrendViewScrollBack, 2313
TrendViewScrollForward, 2312
TrendViewSetRulerMode, 2316
TrendViewStartStop, 2317
Trigger tag, 2168
Triggering by means of a configured function
Transferring data, 2454, 2514, 2553, 2689
Triggers, 2377
Event trigger, 2377, 2380
TRUNC, 1347, 1519, 1626
TS adapter, 59
TS Adapter
Installing additional software, 3101
TS Adapter IE
Brief description, 3109
Default parameter assignment, 3111
parameter Assignment, 3114
Principle of operation, 3110
TS Adapter IE Basic
Connection to the GSM network, 3112
Connection to the telephone network, 3112
Connection to the telephone network through an external modem, 3113
Connection types, 3111
TS Adapter MPI
Brief description, 3101
Default parameter assignment, 3103
Direct connection, 3103
Establishing a modem connection, 3104
Establishing the direct connection, 3104
Exporting adapter parameters, 3108
Importing adapter parameters, 3108
Modem connection, 3104
parameter Assignment, 3106
Parameter assignment, 3106
Parameter assignment options, 3105
Principle of operation, 3102
Restoring the default parameter assignments, 3107
Setting up access protection, 3117
TSAP
ASCII code table, 411
Structure, 381, 409
TSAP assignment
Examples, 411
Twos complement, 1296, 1466
Type
Library, 2072
Types of DP slave, 473

U

UART data transmission, 560
UDINT, 754, 792, 809
UDP
Characteristics, 403
Port numbers, 407
UFILL_BLK, 1336, 1508, 1612
UINT, 752, 790, 807
UMOVE_BLK, 1332, 1503, 1608
Underflow, 609
Undoing actions
Basics of undoing, 257
Undoing last action, 259
Undoing multiple actions, 259
Uniqueness
of object names, 115
Universal
PROFIBUS, 2440, 2491
Universal Serial Interface Protocol, 1914
unlock
Users, 2271
Unplugged module, 332
Unspecified CPU, 338
Update
Tag, 2136, 2142
Update cycle, 2154

Update time, 509
UpdateTag, 2302
Updating
    Operating system, 2781
Updating the operating system, 2780
Use, 2221
    Of recipes, 2221
    Screen navigation function keys, 2062
    simple alarm view, 2832
    simple alarm window, 2832
    Simple user view, 2823
    System function, 2286
Use global assignment
    Function key, 1979
Useful information on configuring the TS Adapter MPI, 3105
User administration, 2251
    Migration, 132
    Object with access protection, 2274
    Runtime settings, 2256
    Setting up, 2277
User change, 2363, 2379
User data
    Area, 540
    Backing up, 2822
    Restoring, 2822
User diagnostic alarm
    Creating, 1190
    Deleting, 1191
    Editing, 1191
User group, 2821
    Administer authorizations, 2263
    Assigning, 2282
    Assigning users, 2262
    Change displayed name, 2264
    Changing in runtime, 2271
    Changing the name, 2264
    Creating, 2281
    Deleting, 2264
    Managing, 2263
    Unauthorized, 2272
User interface
    Details view, 189
    Inspector window, 184
    Maximizing the work area, 177
    Minimizing the work area, 177
    Overview window, 190
    Portal view, 167
    Project tree, 171
    Project view, 169
    Resetting the user interface layout, 193
    Task card, 186
    Views, 166
    Work area, 175
User interface language, 2723
    Selecting, 2726
User program
    Finding errors, 1008
    Function, 694
    Testing, 1127
User view, 2266
    Application, 2085
    Configuring, 2267
    Layout, 2085
    Number of lines, 2086
    Permitted characters, 58
    Simple user view, 2266
User-defined functions
    In function lists, 2288
User-defined text lists
    Creating, 266
    Editing, 267
    Editing value ranges and texts, 267
Users, 2822
    Admin, 2828
    Assigning a user group, 2262
    Changing, 2270
    Changing the name, 2262
    Changing, touch operation, 2827
    Creating, 2261, 2282, 2824
    Creating in runtime, 2268
    Creating, touch operation, 2824, 2826
    Deleting, 2263, 2828
    Deleting in runtime, 2270
    Logging logons, 2276
    Logoff, 2825
    Logon, 2271
    Managing, 2262, 2269
    Unlock, 2271
    Updating after change of user, 2379
Using a TS Adapter for TeleService, 3100
Using frames and cover pages from the library, 243
Using tags, 2087
Using the keyboard
    Program editor, 867
Using the on-screen keyboard, 197
USINT, 750, 788, 804
USS communication
    Controlling a transmission to a drive with USS_PORT, 1918
    Data exchange with drives via USS_DRV, 1919
    Modifying parameters in the drive with USS_WPM, 1923

Reading parameters from the drive with USS_RPM, 1922
USS protocol
Instructions, 1914
USS status codes, 1925

V

Valid
Data type, 2567, 2613, 2623, 2639, 2651, 2668
Data types, 2460, 2461, 2519, 2560
Valid data type
Allen-Bradley DF1, 2594
Value, 1469
Value assignment, 981
Value change, 2372
Value/Range
Text list, 2025
VARIANT, 773
Vertical movement
Animation, 2044
Vertical radius, 2090
View options
For the assignment list, 1098
Setting the call structure, 1108
Setting the dependency structure, 1114
Views of the cross-reference list, 1122, 2720
VOID, 774

W

Warm restart, 528
Watch table
Basic mode, 1134
Copying, 1138
Creating, 1137
Example of filling out a watch table, 1140
Expanded mode, 1134
Layout, 1134
Loading data blocks during an active control job, 42
Meaning of the columns, 1134
Meaning of the icons, 1136
Modify with trigger, 41
Monitoring and modifying modes, 1145
Multiple access to the same CPU, 41
Opening, 1138
Overview of the display formats, 1143
Overview of the test options, 1133
Permitted operands, 1141
Permitted operands for modify values, 1141
Possible applications, 1133
Rounding floating-point numbers, 41
Saving, 1139
Switching between basic mode and expanded mode, 1135
Syntax check, 1140
Testing wiring, 1133
Watchdog time, 509
Web application, 439
Web Control DB, 456
Web pages in browser, 462
Web server, 435, 436, 439
Enable, 437
HTTPS, 438
When dialog is closed, 2362
When dialog is opened, 2362
WHILE, 1638
WinCC flexible project
Migrating, 117, 136
WinCC flexible Runtime
Closing, 2793
Wire break, 609
Wireless LAN/PB link, 490
Wiring tests, 1133
Wizard
Device wizard, 2692
WORD, 747, 785, 801
WORD_TO_BLOCK_DB, 819
Work area
Embedding floating elements, 179
Floating elements, 179
Function, 175
Maximizing, 177
Maximizing elements, 182
Minimizing, 177
Minimizing elements, 182
Saving a layout of editors and tables, 183
Splitting, 179
Switching between elements, 182
Using grouped elements, 180
Work memory, 535, 1117
Working step
to create screens, 1974
Working with the phone book, 3122
WRIT_DBL, 41
Writing data
in remote CPU with PUT, 1856
To DP standard slaves/PROFINET IO devices with DPWR_DAT, 1723
Writing tags, 447
WWW (instruction), 455
WWW Synchronizing user-defined web pages, 1893

X

xlsx file, 2699, 2707, 2711, 2712
XOFF, 38
XON, 38
XOR, 1372, 1396, 1397, 1545

Z

Zoom
  Adjusting the zoom setting, 308, 311, 314
  Keyboard shortcuts, 324
Zoom in/out, 324
Zooming
  Screen, 1973